Reconocimiento de caracteres mediante imágenes en contadores de gas en entornos reales

Ignacio Hidalgo Bejarano
Roberto José Sánchez García de Blas

Madrid, junio de 2015

Proyecto de la Facultad de Informática
Departamento de Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid

Trabajo Fin de Grado en Ingeniería Informática e Ingeniería del Software
Curso 2014-2015

Director: Gonzalo Pajares Martinsanz

Una computadora puede ser llamada "inteligente" si logra engañar a una persona haciéndole creer que es un humano.

Alan Mathison Turing (1912-1954)

Autorización de difusión y utilización

Los abajo firmantes, matriculados en el Grado de Ingeniería del Software impartido por la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo de Fin de Grado, realizado durante el curso académico 2014-2015 y bajo la dirección de Gonzalo Pajares Martinsanz en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet, y garantizar su preservación y acceso a largo plazo.

Ignacio Hidalgo Bejarano Roberto José Sanchez Gª de Blas

Madrid, junio de 2015

Agradecimientos Personales

Antes de nada queremos dedicar unas palabras de agradecimiento en término particular.

Ignacio Hidalgo

Me gustaría recordar en especial a mi madre, sin ella seguramente no estaría escribiendo estas líneas. Me lo has dado todo y has conseguido que todo lo que haga lo haga pensando en ti, gracias Mamá. También se lo quiero agradecer al resto de mi familia, a mi padre, gracias.

A mis amigos, ellos lo son todo, las personas que más me han ayudado siempre y que nunca tendré suficientes palabras de agradecimiento.

A Roberto, por haber podido realizar este proyecto juntos, pero sobre todo por haber encontrado un amigo para toda la vida.

Gracias a todo el mundo que en algún momento ha logrado que mejore, GRACIAS.

Roberto Sánchez

A mis padres, gracias por el esfuerzo, la confianza y la fe ciega que habéis depositado siempre en mí. La libertad que me habéis ofrecido para poder elegir mi propio camino me ha permitido llegar hasta este punto de inflexión en mi vida. Tengo la enorme suerte de seguir aprendiendo junto a vosotros cada día.

A mi hermano, cuya etapa universitaria comienza donde acaba la mía y del cual me siento muy orgulloso.

A ese grupo de compañeros de la universidad, por convertir estos años de carrera en la mejor etapa de mi vida, y en especial a mi amigo Nacho, que firma este proyecto conmigo, con el cual he vivido buenos y malos momentos, y que espero que a partir de ahora sean todos buenos porque se lo merece.

A mis amigos, que han colaborado en este proyecto de la mejor manera, ayudándome a desconectar.

A todos vosotros, GRACIAS.

Agradecimientos

A Madrileña Red de Gas por su confianza depositada en nosotros para llevar el proyecto a cabo, en particular D. Glen Lancastle, cuya clara exposición de los requisitos en las reuniones iniciales, nos ayudó a conocer la problemática y necesidades de la empresa.

Y por último, agradecemos al Departamento de Ingeniería del Software e Inteligencia Artificial de la Facultad de Informática de la Universidad Complutense de Madrid, por proponer este proyecto, y en especial a nuestro director, Dr. Gonzalo Pajares, por sus enseñanzas, dedicación y confianza mostradas.

Índice

Tabla de Figuras

Lista de Abreviaturas

ANPR	*Automatic number plate recognition*
CSS	*Cascading Syle Sheets*
CUPS	*Código Universal de punto de suministro*
EAN	*European Article Number*
HOG	*Histogram of oriented gradients*
HTML	*HyperText Markup Language*
IMR	*Intelligent Machines Research Corporation*
JS	*JavaScript*
LBP	*Local Binary Patterns*
MVC	*Model-View-Controller*
MVT	*Model-Views-Template*
OCR	*Optical Character Recognition*
PIL	*Python Imaging Library*
RGB	*Red Green Blue*
ROI	*Región de Interés*
SaaS	*Software as a Service*
SQL	*Structured Query Language*
SVM	*Support Vector Machines*
TFG	*Trabajo Fin de Grado*
UCM	*Universidad Complutense de Madrid*

Resumen

En los últimos años ha habido una significativa evolución en los teléfonos inteligentes, así como en la calidad de las cámaras que éstos incorporan y en las redes de alta velocidad 3G/4G. Esto ha provocado que las empresas replanteen reducciones en los costes de las infraestructuras tradicionales mediante el uso de estos dispositivos, haciendo el trabajo de sus trabajadores más cómodo y delegando ciertos aspectos a sus propios clientes.

En este proyecto se propone una solución a un problema real de la empresa Madrileña Red de Gas, que mientras prepara el despliegue de contadores inteligentes, necesita un método automático de lectura de sus contadores de gas analógicos mediante una imagen y una aplicación informática capaz de determinar la lectura e interpretación de los dígitos relevantes. Para ello será necesario identificar tanto el código de identificación del contador como la lectura del consumo actual. Esta imagen se toma siempre en un entorno real, sin la necesidad de realizar un encuadre sobre el contador.

Para ello se hace uso de las disciplinas de Aprendizaje Automático, para desarrollar un detector de objetos y el reconocimiento óptico de caracteres mediante la aplicación de técnicas de visión por computador, cuyo objetivo es el procesado y las transformaciones de imágenes. Simultáneamente se propone el desarrollo de tecnologías web cuya finalidad es la integración de los desarrollos del proyecto en una plataforma operativa.

Palabras clave: Contadores de gas, Aprendizaje automático, Detección de Objetos, Segmentación, Visión por computador, Reconocimiento de caracteres, Aplicaciones Web

Abstract

In the last few years there have been significant developments in smartphones, as well as the improvements in quality terms of the cameras that these devices incorporate and the expansion of 3G/ 4G high-speed networks. This has led companies rethink reductions in the costs of traditional infrastructure through the use of these devices, doing the work of their employees more comfortable and delegating certain aspects to their own customers.

This project proposes a solution to a real problem of the Madrileña Gas Network Company, which as it prepares for the deployment of smart gas meters, they need an automatic method of reading their gas meters using a digital image and a computer application that should be capable of determining the reading and interprets the relevant digits. To do that it will need to determine the identification code of the gas meter and the reading of the current consumption. This image is always taken in a real environment, without the need for a frame over the counter.

This makes use of the disciplines of machine learning, to develop an object detector and the optical character recognition through the application of techniques from computer vision, whose aim is processed and the transformations of images. Simultaneously proposing the development of web technologies whose purpose is the integration of the project developments in an operational platform.

Keywords: Gas meters, Machine Learning, Object Detection, Segmentation, Computer Vision, OCR, Web Applications

1. Introducción

El proyecto surge a raíz de una propuesta de la empresa Madrileña Red de Gas, la cual expuso ante nuestro director el problema real con el que se encuentran en relación a la lectura de los contadores de gas bajo su control. El objetivo principal consiste en la automatización del proceso de lectura de los contadores mediante el desarrollo de una aplicación informática. Bajo esta perspectiva, la idea que subyace es doble. Por un lado se trata de agilizar eficientemente el proceso de gestión de lectura de los contadores. Por otro lado, se pretende avanzar en el campo tecnológico, relativo al reconocimiento de caracteres en imágenes, para interpretar y asociar las lecturas del consumo de gas así como su relación con el número de suministro contratado. En definitiva, la tecnología informática al servicio de la gestión en la lectura de contadores de gas.

Madrileña Red de Gas [19] opera en 59 municipios de la Comunidad de Madrid con una red de distribución de 5.533 Km . Presta servicio a diferentes compañías del sector, entre las que desatacan Iberdrola, Endesa, Gas Natural, Galp Energía, etc.

Con el fin de fijar los objetivos de la aplicación a desarrollar, en una primera fase de definición, se realizaron varias reuniones de toma de contacto con responsables de la empresa encabezados por su director de información, D. Glen Lancastle. En el primero de estos encuentros se nos expone el proceso de lectura de los contadores de gas. La descripción del proceso es el que realiza la misma empresa para todas las firmas suscritas a su red. Esta lectura suele realizarla un equipo de profesionales que visitan el contador a lo largo de distintos periodos anuales o bien mediante el envío de los datos relativos a la lectura por parte de los usuarios. En el primer caso los inconvenientes son evidentes, en el sentido de que la vivienda o instalaciones donde se ubican los contadores no son de libre acceso, lo que origina que en muchas ocasiones no pueden realizarse debido a la ausencia del propietario, o por otros impedimentos físicos tales como la existencia de armarios de comunidades sin portero. En el segundo de los casos, queda bajo la responsabilidad del usuario el envío de la información a la empresa, aún a pesar de la buena voluntad por parte de aquellos, es posible que la falta de un procedimiento sencillo impida un envío correcto de la información. En cualquier caso, estas lecturas están lejos de ser automáticas, ya que bien por la parte del profesional o por la parte del cliente, se deben anotar las lecturas manualmente, recoger el CUPS (Código Universal de Punto de Suministro) y enviarlas por procedimientos tradicionales o mediante una aplicación web. En cualquier caso, existe una fuerte dependencia de un proceso manual, que es lo que se pretende resolver mediante el desarrollo de la aplicación que se describe en el presente trabajo.

En la actualidad resulta bien conocido, a través de los medios de comunicación o por experiencia directa, el hecho cierto sobre la posibilidad de cambiar los contadores clásicos de luz a contadores inteligentes. Esa posibilidad también existe para los contadores de gas, aunque probablemente en un futuro más lejano. Con los contadores inteligentes se pretende automatizar todo el proceso relativo a la lectura, a la vez que se monitoriza el consumo de una forma inteligente mediante conexiones o aplicaciones web. En el caso de los contadores de gas, el objetivo primordial consiste en ahorrar las visitas a los domicilios por parte del personal

encargado de la lectura a la vez que se avanza en el proceso de automatización de datos de la mejor manera posible, evitando así un sobrecoste muy alto en el procedimiento de lectura. Por otra parte se evita la necesidad de cambiar todos los contadores de cada casa de forma masiva. La aplicación que se describe en este trabajo está orientada a solventar este problema de una manera suficientemente satisfactoria.

Madrileña Red de Gas ofrece un sistema desde su misma página web para que los clientes introduzcan las lecturas de sus contadores, identificándose mediante su DNI y el CUPS. Con este sistema el usuario puede subir su consumo en su periodo de facturación, sin que la empresa tenga que hacer estimaciones respecto a los meses anteriores. Con el método de las estimaciones es posible llegar a tener problemas a la hora de facturar, ya que el cliente puede tener un consumo no habitual por ejemplo, en periodos de vacaciones o simplemente puede haber introducido datos de lectura erróneos. El problema se resolvería cuando se recoja la siguiente lectura real, pero durante ese periodo el cliente ha podido generar consumos más elevados que pueden obligar a la empresa a realizar ajustes en futuras facturaciones, con el consiguiente perjuicio económico.

Por otra parte, la solución de informar sobre los datos de consumo vía web no resulta totalmente válida, ya que no todos los usuarios habituales conocen el sistema de acceso a la web, además los clientes en ocasiones no saben dónde buscar el código CUPS, la página web no está adaptada a las plataformas móviles y un largo etcétera de inconvenientes derivados. De ahí el interés de Madrileña Red de Gas por buscar una solución que permita al cliente tomar su lectura de la manera más sencilla, tomando una simple fotografía desde su *smartphone* o dispositivo móvil y una aplicación que se encargue de automatizar su lectura y asociarla al cliente previamente identificado en la aplicación con su DNI.

El siguiente paso en la definición y avances del proyecto se concretó en una serie de reuniones posteriores en base a uno de los contadores que la empresa tiene repartidos por toda la Comunidad de Madrid, así como en relación a la idea general de su propuesta. El esquema se fundamenta en el desarrollo de una aplicación software basada en el tratamiento de imágenes, capaz de reconocer la fecha de la toma, la lectura de los metros cúbicos de gas consumidos (se especifica que los decimales no son necesarios a la hora de facturar), y por último el número de contrato asociado al suministro.

Junto a este *backend* debe desarrollarse también un sistema de interacción básico que permita subir las fotografías para su lectura en la fase de testeo, pensando que en un futuro el *backend* ha de ser utilizado desde una aplicación móvil.

Después de los progresos realizados en las primeras fases de desarrollo del proyecto, miembros de la propia empresa se mostraron interesados en tener una reunión con nosotros para conocer de primera mano los planteamientos iniciales de la futura aplicación a desarrollar.

Conviene mencionar en este sentido, que para el desarrollo de la aplicación se parte de cero, es decir en las fases iniciales no se disponía de una base de datos relativa a imágenes de

contadores ni por supuesto a desarrollos previos. Esto ha derivado en el hecho de plantear el problema de reconocimiento de caracteres en los contadores desde la base. Para ello se ha comenzado con el estudio y análisis tanto de los procesos relativos a la captura de imágenes como de los métodos de aprendizaje automático necesarios para alcanzar los objetivos.

El hecho de utilizar métodos de aprendizaje automático, basados en el procesamiento digital de imágenes, para reconocer tanto los caracteres que conforman la lectura del consumo de gas, como el código de usuario nos ha permitido utilizar imágenes de distintos contadores de gas en el proceso de entrenamiento, en los que tanto la ubicación de las numeraciones como el tipo de caracteres varía respecto del que se ha utilizado como base. Esto ha permitido realizar un desarrollo de una aplicación con amplia validez para su uso en la lectura de otros contadores de gas, tanto los utilizados por Madrileña Red de Gas como los procedentes de otras empresas encargadas de la lectura de los citados contadores. Con lo cual nuestro sistema es fácilmente transportable para su uso por otras empresas y con otros contadores, otorgando así una validez los más universal posible al proyecto.

1.1. Objetivos y organización del trabajo

El planteamiento del proyecto se basa en un objetivo real, surgido de la necesidad de automatización de las lecturas de contadores, por parte de la mencionada empresa Madrileña Red de Gas, cuyo ámbito de actuación comprende una parte importante del territorio español. Con lo cual el proyecto implica el desarrollo de una aplicación de utilización práctica en el ámbito empresarial, cuya finalidad es dar solución a un problema real surgido de una necesidad empresarial.

El punto inicial para comenzar el desarrollo del proyecto fue una primera toma de contacto, materializada en una reunión de trabajo, con el director de información de Madrileña Red de Gas, quien expresó sus puntos de vista y planteamientos inherentes al problema en cuestión. En esta primera reunión se determinaron los criterios básicos para el desarrollo de la aplicación, cuyo núcleo central consistía en la captura de imágenes para su posterior tratamiento con fines de identificación de las cadenas de caracteres que constituyen tanto la lectura del consumo de gas como la de identificación de usuario.

Dado que se inició el proyecto desde sus orígenes, el mismo comenzó sin ningún material previo relativo a la base de datos de imágenes, con lo que en estas primeras fases hubo que dedicar más tiempo y recursos a investigación que al propio desarrollo. Más específicamente, la investigación se centró en encontrar y seleccionar los mejores métodos para solucionar el problema planteado.

Tras un minucioso estudio sobre la problemática a resolver y derivaciones asociadas, se planteó el desarrollo de una aplicación para que el cliente se registre en la web de la empresa de forma que con la captura de una imagen frontal del contador, el sistema sea capaz de reconocer tanto

el identificador único de usuario, sirviéndose del código de barras, que está asociado a un único cliente, así como del consumo actual del contador, descartando las cifras decimales.

La aplicación aunque en un principio está pensada para que la utilicen los usuarios de Madrileña Red de Gas, es fácilmente transportable a otros sistemas e incluso con capacidad de aplicación a otros tipos de contadores. También al estar basada en una aplicación web, en un futuro el propietario del contador podría desde un dispositivo móvil realizar la captura de la imagen, procesarla y enviar los resultados de la lectura y código de usuario en remoto.

El objetivo final y principal es desarrollar una aplicación funcional, con la mayor tasa de acierto posible y siempre tratando de superar el umbral del 60%, que constituye un objetivo perfectamente asumible por parte de la empresa en estas primeras fases de desarrollo, con el fin de progresar hacia tasas de aciertos mayores en desarrollos sucesivos Otros objetivos importantes de conseguir son los que se detallan a continuación:

- Disponer de un histórico de las lecturas y de las imágenes procesadas, para que el cliente lo pueda consultar en cualquier momento.
- Desarrollar una aplicación que se pueda escalar para varios dispositivos, para que el usuario pueda subir las imágenes desde cualquier plataforma, preferentemente desde dispositivos móviles.

1.2. Estructura de la memoria

La memoria está distribuida en seis capítulos, claramente diferenciados, en los que se detalla tanto el planteamiento del proceso seguido en la elaboración del proyecto como el desarrollo del mismo. Además, se incluyen cinco apéndices con el fin de proporcionar aspectos de interés relativos a la propia aplicación.

El capítulo 1, realiza una introducción de cómo se plantea y planifica el proyecto. También se explican los objetivos y las contribuciones de cada participante.

En el capítulo 2 se explica el estado del arte relativo al procesamiento de imágenes, orientado al desarrollo de la aplicación. Se comienza con una introducción a la visión computacional y sus aplicaciones. Después se continúa con temas relativos al procesado digital de imágenes y su uso en la aplicación. Más específicamente, se analizan técnicas propias relativas al reconocimiento óptico de caracteres, utilizadas para el reconocimiento de los dígitos relativos al consumo de gas. Para la detección de la zona donde se encuentran los datos de consumo, se han utilizado técnicas propias sobre el reconocimiento de objetos. Otros procedimientos utilizados para situar sobre la imagen esa zona son técnicas de aprendizaje automático. Con todos ellos se pasa de una imagen digital a la grabación de los datos procedentes de la lectura de las secuencias de caracteres que constituyen el objetivo de su identificación.

El capítulo 3 detalla las técnicas utilizadas para el tratamiento de imágenes, incluyendo algunas tales como eliminación de ruido de una imagen, eliminación de bordes o técnicas de binarización.

El capítulo 4 explica las fases preliminares de desarrollo, más concretamente cómo se trata de alcanzar los objetivos finales junto con la elección de todas las tecnologías que se usaron en el proyecto.

El capítulo 5 trata sobre el análisis y especificación de requisitos, los casos de uso, los patrones utilizados para la construcción de la aplicación y los diagramas asociados que ilustran el funcionamiento de la aplicación.

En el capítulo 6 se detallan las tecnologías utilizadas para el correcto funcionamiento de la aplicación, así como las dependencias de módulos externos.

En el capítulo 7 y último se exponen las principales conclusiones del trabajo realizado, así como los resultados obtenidos. También se muestran posibles mejoras que se pueden incorporar en futuras versiones de desarrollo de la aplicación, previstas fundamentalmente para la ampliación de su alcance.

En el Apéndice A se detalla la manera de instalar la aplicación, para distintos sistemas operativos, tales como Windows o Linux.

En el Apéndice B se muestra el despliegue de la aplicación en otro sistema web, con todas las dependencias existentes y los recursos necesarios para poder portarlo.

El Apéndice C es en realidad un manual de uso para el usuario final, en el que se explican los casos de uso o funciones básicas de la aplicación.

El Apéndice D aclara el método para la detección de la zona del número de serie del contador, mediante tratamiento de imágenes.

El Apéndice E detalla el proceso de entrenamiento desarrollado para la detección de la región de interés que delimita la zona de los dígitos en el contador. En este caso sin técnicas de aprendizaje automático.

El Apéndice F detalla el proceso de entrenamiento desarrollado para la detección de la región de interés que delimita la zona de los dígitos en el contador, aunque en este caso mediante técnicas de aprendizaje

El Apéndice G se detalla el proceso de entrenamiento, para la detección de zona del contador de gas.

1.3. Contribuciones personales

1.3.1. Ignacio Hidalgo Bejarano

Al inicio de este trabajo, lo primero que hice fue asistir a la reunión con el representante de la empresa Madrileña Red de Gas, D. Glen Lancastle, donde tomé las primeras anotaciones sobre los objetivos que se querían alcanzar a la finalización del proyecto.

Después de ver la meta que se quería lograr con esta aplicación, realicé las primeras fotografías para iniciar una fase previa de pruebas relativas al tratamiento de imágenes. En estas primeras pruebas, el objetivo fundamental consistió en detectar una zona característica del contador, identificada por una tonalidad roja. La identificación de dicha zona se basó en la aplicación de técnicas de procesamiento de imágenes basadas en la segmentación por color. Seguidamente, se estudió la posibilidad de detección de la parte numérica contigua a dicha zona para proceder a la lectura de los dígitos que determinan el consumo de gas. En todo este proceso se aplicaron técnicas de tratamiento de imágenes encaminadas a limpiar la imagen resultante de la segmentación de todo el ruido subyacente.

En cuanto al análisis de imágenes, aunque había cursado una asignatura llamada Percepción Computacional, impartida por mi director del TFG (Trabajo Fin de Grado), se basaba en su amplia mayoría en el tratamiento de imágenes. Tenía una diferencia circunstancial ya que en dicha asignatura se utilizaba MATLAB, con su propio lenguaje (lenguaje M), mientras que en el presente proyecto la librería utilizada es OpenCV y aunque esencialmente el concepto relativo al tratamiento de imágenes es el mismo, la manera de materializar y aplicar las rutinas proporcionadas por MATLAB y OpenCV no es exactamente igual.

A lo largo de esas pruebas empezamos a dividir más el trabajo entre las personas que formamos el equipo de desarrollo, de esta forma podíamos diferenciar varias funciones para cada miembro del equipo. A partir de ahí, yo me dedique más en profundidad al desarrollo de la aplicación web y mi compañero se dedicó más específicamente al tratamiento de imágenes.

En lo que respecta a la aplicación web antes mencionada, me he encargado de crearla a partir de cero, esto llevó una curva de aprendizaje alta, ya que aunque había utilizado alguna vez Python y el framework Django, no había sido al nivel que necesitábamos para materializar el desarrollo de la aplicación. El primer punto sobre el que me centré era poder tener una aplicación disponible para distintos sistemas operativos, con lo que creé un entorno virtualizado con *virtualenv*, en el cual se incorporan los paquetes independientemente del sistema operativo que se utilice.

Como paso previo al desarrollo de la aplicación se decidió entre los componentes del grupo tener la aplicación en un repositorio para poder trabajar de manera conjunta, en un primer momento se iba a utilizar GitHub, por su gratuidad, aunque los repositorios son públicos y pueden estar al alcance de todo el mundo. Con bitbucket este problema se solucionaba ya que es gratuito para

grupos de menos de 5 personas y aparte es privado, con lo cual solo teníamos acceso los componentes del equipo. Creé el repositorio, con la aplicación inicial, para empezar a trabajar a partir de ahí.

Una vez creada la aplicación con las librerías necesarias, lo primero que hice fue integrar toda la parte visual para el usuario, en la que se incorpora toda la zona de usuario y la plataforma de subida de imágenes. Al principio lo primero que generé es la página de subida de imágenes, aunque el usuario no estuviera registrado en la aplicación, ya que esto era el objetivo principal del proyecto. Es decir, que se pudiera subir una imagen y que se detectaran tanto el código identificativo de contador como la lectura de consumo actual. Este punto era el más complejo dentro del desarrollo planteado, ya que a través de una imagen subida a la aplicación, posteriormente la librería de OpenCV era la que tenía que interpretarla, tras la aplicación de las técnicas de procesamiento y reconocimiento de imágenes. En un primer momento esto produjo una serie de errores ya que la imagen debía estar guardada en la propia base de datos de la aplicación. Todo esto llevó a usar los modelos de Django, en el que todas las imágenes subidas se guardan en una base de datos SQLite que permite poder llevar a cabo el tratamiento el tratamiento requerido así como la recuperación de las imágenes procesadas en etapas posteriores.

Más adelante, cuando ya estaba planteada y desarrollada toda la parte relativa a la subida de imágenes, me encargué del sistema de gestión de los usuarios, que engloba el registro de usuarios en la web. La gestión de usuarios implica la creación de la zona web, del sistema de base de datos para guardar los datos y la lógica de la gestión.

Una vez avanzado en cuanto al tratamiento de imágenes y la gestión de usuarios se refiere en el equipo decidimos añadir alguna mejora funcional a la página, tales como el histórico de imágenes del usuario o que en vez de guardar los datos de manera automática, el usuario pudiera visualizarlos para poder confirmarlos.

La persistencia de datos también la desarrollé conjuntamente con la aplicación web. Esta parte está dividida en dos sistemas, uno para guardar el registro de los usuarios y otro para los datos de consumo, estos documentos se guardan en MongoDB, que es un sistema de bases de datos NoSQL. Las imágenes que suben los usuarios se guardan en SQLite, cuya principal característica estriba en el hecho de que se trata de un sistema relacional.

Además del desarrollo relativo a la parte de la aplicación basada en el aprendizaje automático, me encargué de generar los ficheros de clasificación de cascadas Haar, para que la aplicación detectara mediante aprendizaje automático la zona donde se ubica los dígitos del contador. Este procedimiento se fundamenta en una tecnología puntera relativa a la detección facial en imágenes digitales, algo de gran utilidad en el desarrollo de la aplicación que se propone en el presente proyecto. Siguiendo con esta línea, se realizaron dos pruebas de forma local en un computador equipado al afecto. La primera se realizó con una serie de ejemplos positivos y negativos, tal y como requiere el método de detección facial mencionado y aunque los resultados fueron aceptables, ya que detectaba la zona del contador en bastantes casos, vimos que los

resultados eran manifiestamente mejorables. En aras de la mejora prevista, se realizó una prueba intermedia que se ejecutó en un ordenador alquilado de Amazon. En la tercera prueba y a la postre la definitiva, se ajustaron más los valores relativos al entrenamiento, incorporando más imágenes tanto positivas como negativas, con lo cual los resultados mejoraron considerablemente con respecto a los anteriores y se decidió que era una solución bastante óptima para los objetivos planteados inicialmente.

Con respecto a la realización de la memoria, nos hemos distribuido el trabajo de manera bastante pareja, me encargué del punto relativo a las técnicas y metodologías aplicadas, más en concreto del análisis de imágenes y su tratamiento morfológico. También realicé la parte correspondiente al análisis, diseño e implementación de la propia aplicación. Obviamente, en este trabajo se incluyen las aportaciones relativas a las partes comunes de la memoria, incluyendo la introducción, conclusiones y trabajo futuro.

1.3.2. Roberto José Sánchez García de Blas

La labor principal de mi contribución al proyecto se centró mayoritariamente en el análisis de las imágenes del contador. Al tratarse de un proyecto que parte de cero, inicialmente se asignó un tiempo a la toma de contacto con la empresa, para establecer los requisitos mínimos que debería tener la aplicación software. Tras este proceso, se realizó un arduo trabajo de investigación de soluciones que pudieran acercarse a lo que se pedía y al mismo tiempo familiarizarme con las herramientas y operaciones propias de las técnicas relativas de visión por computador.

La metodología se centró en dividir la información que necesitábamos encontrar en la imagen, y establecer una serie de posibilidades para obtener dicha información. Para cada alternativa se desarrollaban una serie de pruebas de concepto con una pequeña interfaz gráfica para la subida de imágenes e interacción con los parámetros asignados. Estas pruebas de concepto eran testeadas en las reuniones de equipo, en las que el director del proyecto podía verificarlas o guiarnos en la búsqueda de soluciones más sencillas o con mayor tasa de acierto.

Tras la elección de optar por lectura de código de barras para el CUPS y de un motor de reconocimiento de caracteres, conocido como OCR (Optical Character Recognition), para la lectura de los metros cúbicos de gas consumidos, se realizó una batería de pruebas para la amplia gama de soluciones software encontradas.

Para el código de barras se probaron zBar, ZXing y Scandit. De los cuales solo zBar y Scandit podían leer el código de barras del contador, que pertenece al estándar EAN18 (European Article Number) , eligiendo el primero, ya que la solución de Scandit era de código cerrado y comercial. Respecto a los motores OCR se probaron dos, Tesseract-OCR y Abbys Cloud OCR SDK, la solución de Abbys proporcionaba mejores resultados sin necesidad apenas de tratamiento en la imagen, pero solo se pudo probar durante el periodo de prueba ya que es un software comercial de pago. Por otro lado, Tesseract-OCR tiene una comunidad amplia, permite reentrenar sus diccionarios y al ser de código abierto es fácilmente integrable en otros sistemas.

La detección de las regiones de interés fue, sin duda, el trabajo más complejo. La detección de la zona del código de barras está basada en la búsqueda de matrículas, en ésta se aplican las técnicas de búsquedas del gradiente X (líneas verticales) a través del método de Sobel. Para la detección de la zona de los dígitos de la lectura, se comenzó con la detección de la región roja de los decimales como primera aproximación, el problema es que el óxido y las tuberías de cobre que llegan normalmente a los contadores de gas tienen una fuerte componente roja, lo cual nos daba regiones de falso positivo, dado que una parte clave en la identificación de la lectura de consumo se basa en la detección de una zona roja característica del tipo de contadores utilizados. Tras la lectura del libro Practical Python and OpenCV, contacté con el Dr. Adrian Rosebrock de la Universidad de Maryland, para buscar una alternativa más fiable. En dicha conversación propuso unas pautas, las cuales implicaban un detector de objetos, y después aplicar transformaciones morfológicas para detectar el cuadrado que encapsula los dígitos. Mi compañero realizó un entrenamiento inicial, basado en las mencionadas cascadas de Haar. No obstante, esto fue problemático ya que las proporciones de los casos positivos no eran óptimos

y estaban basados en un conjunto relativamente reducido de imágenes reales. Pero a pesar de estos fallos, se observó que el detector funcionaba bastante bien, eso sí, el clasificador devolvía una imagen más grande que la esperada debido a los problemas de proporción antes comentados.

Visto que el tiempo del primer entrenamiento se demoró varios días en un Macbook Pro de alta gama, para el segundo entrenamiento alquilé una instancia EC2 de Amazon basada en Linux que dispone de un Intel Xeon E5-2680 8 núcleos y 15 gigas de memoria RAM. En este segundo entrenamiento, obtuve un sobreajuste por la enorme cantidad de casos positivos generado, y tuvo que ser desechado. El tercer entrenamiento fue el definitivo, ya que se aplicaron las proporciones adecuadas en la generación de casos positivos, aplicando transformaciones y rotaciones tridimensionales sobre las imágenes y se ajustó lo suficiente la frontera de detección. Los casos base de los ejemplos positivos fueron recortados con el software ImageClipper, se introdujeron casos base de contadores de otras compañías para que el clasificador fuera más versátil al igual imágenes donde se encontrase humedad de lluvia, reflejos de flash, etc.

Paralelamente a la aplicación de Aprendizaje Automático en la región de lectura, desarrollé un detector de regiones basado tan solo en técnicas de visión por computador, para que se permitiese el reconocimiento parcial en modelos de contador no tratados en el aprendizaje e incluso poder ser utilizado en otro tipo de contadores de suministros.

Tras obtener regiones válidas, debido a que el software Tesseract-OCR es bastante sensible a los elementos que no son estrictamente los dígitos, se pasa una lista de caracteres que debe buscar, en este caso “0123456789” y se aplican operaciones morfológicas y de detección de contornos para obtener exactamente los 5 primeros dígitos, eliminando de la región los decimales, que no constituyen un elemento esencial en la detección. Para la mejora de los resultados obtenidos, comencé a trabajar en el proceso de reentrenamiento de los diccionarios, añadiendo tomas de caracteres reales, junto a los casos conflictivos (diales del contador que están entre dos dígitos) con el software jTessBoxEditor, pero por problemas de calendario no fue posible su incorporación en el software final.

Durante la última fase de desarrollo y para aprovechar la infraestructura desplegada, desarrollé una aplicación para dispositivos Android, la cual representa una previsualización de la cámara, que periódicamente busca el contador haciendo uso del código de barras. Al detectarlo para el proceso de previsualización, y envía la captura tomada mediante una llamada POST al servidor Django. Gracias a la arquitectura de la aplicación, no se requirió de ninguna modificación adicional en el software desarrollado previamente. El servidor, tras procesar la imagen, devuelve como respuesta a la aplicación el código de identificación y la lectura, mostrándolas en una ventana emergente. Para poder usar la aplicación en cualquier lugar, desplegué el servidor Django en una máquina EC2 proporcionada por Amazon gratuitamente. En dicho despliegue pude comprobar de primera mano la escalabilidad y el carácter multiplataforma de nuestro software, que al estar residiendo en la nube, puede comenzar a utilizarse en usuarios finales, cumpliendo con el modelo de distribución de SaaS (Software como servicio).

En cuanto a la realización de la memoria, al haber estado mi parte más enfocada a la investigación, me he encargado del estado del arte, las fases previas del desarrollo y la construcción de los apéndices, donde se muestra con detalle los diferentes procesos que se han llevado a cabo. El resumen, la introducción y las conclusiones al tratarse de temas comunes se han realizado conjuntamente.

Introduction

The project comes as a proposal of the Madrileña Gas Network Company, they presented to our director the real problem for reading gas meters under their control. The main goal consists in the automation of the process of reading the counters through the development of a computer-based application. Under this perspective, there are two main goals. On the one hand it is efficiently streamline the management process of reading the counters. On the other hand, the goal is to make progress in the technological field, relative to the character recognition in images, in order to interpret and associate the readings of gas consumption as well as its relation to the number of contracted supply. In short, the computer technology at the service of the management in the reading of gas meters.

Madrid Gas Network [18] operates in 59 cities in the Autonomous Community of Madrid with a distribution network of 5,533 km. Provides service to various companies in the sector, like Iberdrola, Endesa, Natural Gas, Galp Energia, etc.

With the aim of setting the goals of application to develop, in a first phase of definition, there were several meetings with people in charge in the company headed by its Information Technology Director, Mr. Glen Lancastle. In the first of these meetings we exposed the process of reading the gas meters. The description of the process is one that performs the same company for all signatures subscribed to their network. This reading is usually performed by a team of professionals who visit the counter over various periods per year or by sending the data relating to the reading by the users

It is now well known, through the media or by direct experience, the fact of changing the analogue meters to smart meters. This possibility also exists for the gas meters, although probably in a more distant future. With the smart meters the entire process is to be automated relating to the reading, at the same time that watch the consumption in a smart way through connections or web applications. In the case of gas meters, the ultimate goal is to save the visits to their homes by some of the staff responsible for the reading and at the same time moving into the process of automation data in the best possible way, thus avoiding an extra cost in the procedure of reading. On the other hand, it avoids the need to change all the counters of each house on a massive scale. The implementation described in this work is aimed to solve this problem in a satisfactory manner.

Madrid Gas Network offers a system inside its own web page for costumers to report the readings of their counters, identified by its ID or the UCPS (Universal Code of Point of Supply. With this system the user can raise their consumption in your billing period, without the company having to make estimates of the previous months.

The solution to report the consumption data via web it is not fully valid, since not all regular users are familiar with the system, in addition the costumers sometimes do not know where to look for the code UCPS. Also, the web page is not adapted to the mobile platforms. So the main interest of Madrileña Gas Network is to look for a solution that would enable the client to take its reading

of the simplest way, taking a simple photo from their smartphone or mobile device and upload to an application that automates the reading process and associates the reading with the costumer.

The next step in the definition and progress of the project was made concrete in some meetings based on a specific model of the different gas meters that the company has distributed across the Community of Madrid. The schema is based on the development of a software application based on the treatment of images, able to recognize the date of the input, the reading of the cubic meters of gas consumed (the decimal numbers are not required), and finally the contract number associated with the meter.

Besides this backend, an interaction system should be also developed, in order to be able to upload specific images during the testing time, thinking that in a future the backend has to be used from a mobile application.

It is worth mentioning in this regard, that we start the development of the application from scratch, i.e. during the initial stages we did not have any database containing images from gas meters nor previous developments neither. This has resulted in the fact to pose the problem of character recognition on the gas meters from the base. For this has been started with the study and analysis of both the processes relating to the capture of images and the methods of automatic programming required to achieve the goals.

Machine learning methods have been used, based on digital images processing, to recognize both the characters that make up the reading of the gas consumption and the user code, it has allowed us to use images of different gas meters in the training process, in which both the location of the strip and the typography of characters vary with respect the ones used as a base. This has allowed a development of an application with wide validity for its use in the reading of other gas meters, both used by Madrid Gas Network as those from other companies. So, our system is easily portable for its use by other companies and with other accountants, thus giving validity, the more universal as possible, to the project.

2. Estado del Arte

En este capítulo se estudian los conceptos que han sido utilizados para la realización del proyecto, un repaso sobre la evolución de la visión por computador, el tratamiento de imágenes y el aprendizaje automático utilizado tanto para la lectura de caracteres como para la búsqueda de las diferentes regiones de interés dentro de las imágenes analizadas.

2.1. Introducción a la Visión por Computador

La Visión por Computador, también conocida como Visión Artificial, es un campo de la inteligencia artificial que incluye una serie de métodos para adquirir, procesar, analizar y llegar a la comprensión del contenido de las imágenes. El propósito general es que un computador sea capaz de recibir información de la escena tridimensional en forma de imágenes a través de una serie de sensores, básicamente cámaras de captura de imágenes, bien estáticas o secuencias de vídeo. En ellas se contiene información numérica o simbólica que será utilizada para el objetivo propuesto según las diferentes aplicaciones. El principal objetivo es realizar una aproximación a la visión humana, incluyendo sus habilidades y capacidades, si bien desde un punto de vista automático comprendiendo de este modo un proceso de percepción computacional o de máquina que permita el análisis de las imágenes de cara a su análisis y comprensión final[15]. En la Figura 1, vemos la relación de la Visión por computador con otras disciplinas.

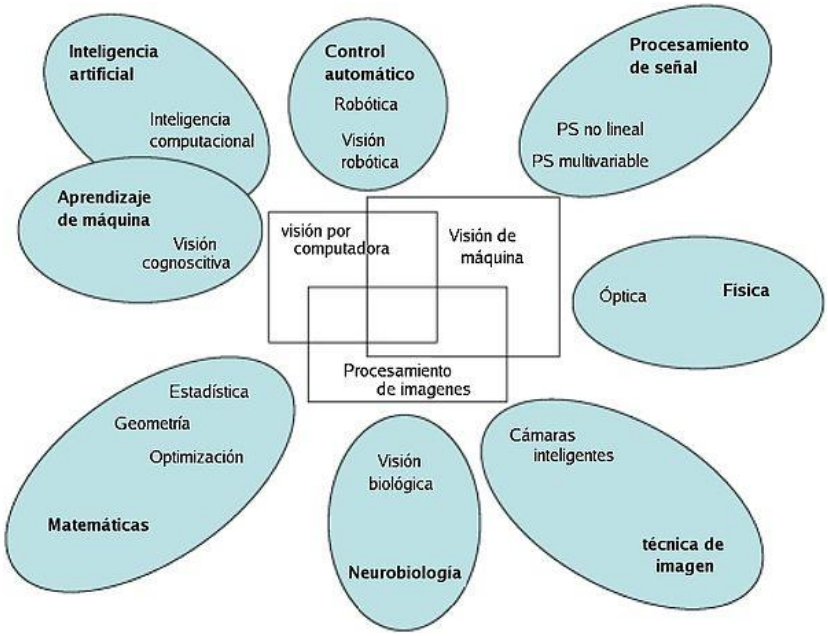


Figura 1: Relación de la visión por computador junto a otras materias [37]

Existen diversas áreas de aplicación de la Visión por Computador, destacando entre otras medicina, agricultura, vigilancia y seguridad, biometría, robótica. Dentro de cada una de ellas se aplican técnicas propias de esta disciplina tales como:

- Reconocimiento y seguimiento de objetos o personas.
- Detección del movimiento
- Reconstrucción 3D

2.2. Tratamiento y procesado digital de imágenes

Es el conjunto de técnicas englobadas dentro del preprocesamiento de imágenes cuyo objetivo fundamental es obtener, a partir de una imagen origen, otra final cuyo resultado sea más adecuado para una aplicación específica mejorando ciertas características de la misma que posibilite efectuar operaciones para el posterior análisis y extracción de información.

Se usan con distintas finalidades, destacando entre otras la detección de bordes, la segmentación de imágenes mediante el reconocimiento de líneas rectas y curvas, análisis del histograma, mejora del contraste, eliminación del ruido o tratamiento de regiones mediante el uso de operadores morfológicos.

2.3. Reconocimiento óptico de caracteres

El reconocimiento óptico de caracteres comúnmente conocido por el acrónimo OCR, es el proceso dirigido a la digitalización de textos, los cuales se identifican automáticamente a partir de imágenes patrón de símbolos o caracteres que pertenecen a un determinado alfabeto, para luego almacenar sus propiedades de identificación en forma de datos. En la Figura 2 se muestra la salida obtenida de un texto de caracteres latinos sin tratar.

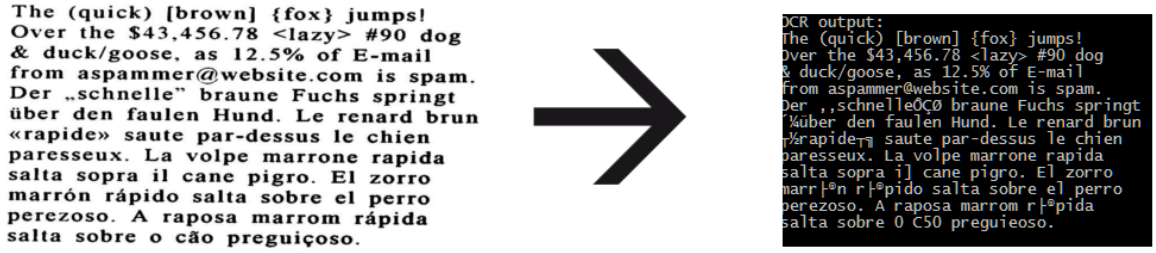


Figura 2: Reconocimiento de caracteres latinos

En 1929, Gustav Tauschek obtuvo una patente sobre OCR en Alemania, luego, Handel [36] en 1933 obtiene la patente de OCR en EEUU. En 1935, a Tauschek [31] también se le concedió una

patente en EEUU por su método. La máquina de Tauschek era un dispositivo mecánico que utilizaba plantillas. Un fotodetector era colocado de modo que cuando la plantilla y el carácter a reconocer estuvieran alineados, una luz era dirigida hacia ellos. En 1950, David Shepard, criptoanalista en la agencia de seguridad de las fuerzas armadas de los Estados Unidos, fue consultado por Rowlett Franco para trabajar con el Dr. Louis Tordella, con el fin de recomendar los procedimientos de la automatización de los datos de la agencia. Esto incluía el problema de convertir mensajes impresos en lenguajes para almacenarlos en un computador. Shepard decide que es posible construir una máquina para realizar ese proceso, y, con la ayuda del cocinero de Harvey, un amigo, construyeron Gismo durante las tardes y fines de semana. Este suceso fue divulgado en los periódicos Washington Daily News y el New York Times en el año 1953, después de que su patente fuera concedida [16]. En este momento, Shepard fundó IMR, comenzando a fabricar el primero de varios sistemas OCR usados para operaciones comerciales. Gismo y los últimos sistemas de IMR, utilizaron análisis de imagen, que comparaban con el carácter a emparejar, pudiendo aceptar una cierta variación de la fuente. Gismo estaba limitado a los registros verticales, mientras que los reconocedores posteriores de la compañía IMR (Intelligent Machines Research Corporation) analizaban caracteres en cualquier parte del campo de exploración, una necesidad práctica en documentos del mundo real.

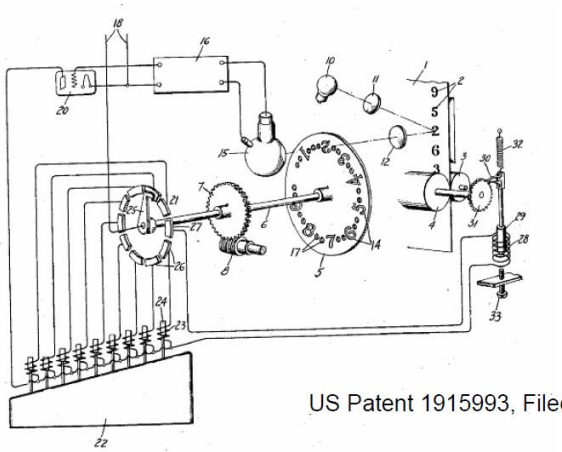


Figura 3: Patente US 1915993 [36]

El primer sistema comercial fue instalado en Readers Digest en 1955, que, muchos años más tarde, fue donado al museo Smithsonian, donde se exhibió. El segundo sistema fue vendido a los Standard Oil Company de California para leer impresiones en tarjetas de crédito con el fin de desarrollar un proceso de facturación automática, además se vendieron muchos más sistemas a compañías petroleras. Otros sistemas vendidos a la fuerza aérea de los Estados Unidos por el IMR durante los últimos años 50 incluyeron un escáner de páginas para la lectura y transmisión de mensajes escritos a máquina. IBM y otras empresas fueron licenciadas más adelante sobre las patentes del OCR de Shepard.

El servicio postal de Estados Unidos ha estado utilizando las máquinas de OCR para clasificar el correo desde que 1965, basados en la tecnología ideada sobre todo por el prolífico inventor Jacob Rabinow. El primer uso del OCR en Europa se utilizó en la oficina de Gran Bretaña. En

1965 Nacional Giro comenzó a utilizar un sistema de actividades bancarias completo, usando la tecnología del OCR, ideó un proceso que revolucionó los sistemas del pago de cuentas en el Reino Unido. El correo postal de Canadá ha estado utilizando sistemas OCR desde 1971. Los sistemas OCR leen el nombre y la dirección del destinatario, e imprimen un código de barras en el sobre basados en el código postal del mismo. Después, las cartas necesitan solamente ser clasificadas por los compaginadores haciendo que los procesos de clasificación sean menos costosos ya que necesitan leer solamente el código de barras. Para evitar interferencias con el campo de dirección escrita a mano, que se puede situar en cualquier parte de la carta, se usa una tinta especial leída bajo la proyección de una luz ultravioleta. Esta tinta parece anaranjada en condiciones normales de la iluminación. Los sobres marcados con el código de barras que son leídos por la máquina pueden ser posteriormente procesados. [2]

El reconocimiento exacto de la escritura latina, ahora se considera en gran parte un problema solucionado. La precisión alcanza ratios del orden del 99%, aunque hay veces en las que se exige incluso una precisión más elevada, requiriendo la revisión humana para subsanar los errores. Actualmente está en desarrollo el reconocimiento de la mano que escribe, al igual que el reconocimiento del texto impreso en otras lenguas, especialmente en las que tienen un número muy alto de caracteres).

2.4. Reconocimiento facial o de objetos

El reconocimiento facial de forma particular o de objetos en general es la tarea encaminada a encontrar e identificar estructuras básicas que definen dichos objetos en una imagen o secuencia de video. Los humanos reconocemos una multitud de objetos en imágenes con poco esfuerzo, a pesar del hecho de que la imagen del objeto puede variar según los diferentes puntos de vista, mostrándose en diferentes tamaños o escalas e incluso rotadas o trasladadas. Los objetos pueden ser reconocidos cuando están parcialmente obstruidos desde una vista. No obstante esta tarea es un desafío para los sistemas de visión por computador. Para resolver este problema se han desarrollado e implementado diversos métodos desde hace ya varias décadas. A continuación se describen brevemente algunos de los más relevantes.

2.4.1. Método de Viola Jones de cascadas Haar

El algoritmo creado por Paul Viola y Michael Jones en 2001 [34], se basa en la extracción de características a partir de lo que se conoce como imágenes integrales mediante las máscaras de Haar. En estas imágenes integrales el objetivo consiste en, considerar regiones rectangulares adyacentes, Figura 4, en una ventana de detección, y sumar la intensidad de los píxeles en cada región, calculando después la diferencia entre estas sumas. Esta suma permite clasificar distintas secciones de la imagen.

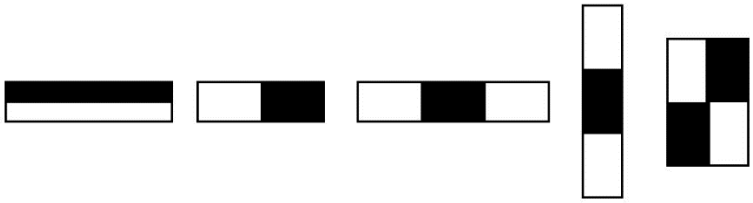


Figura 4: Ventanas de detección [34]

Por poner un ejemplo, si imaginamos que estamos tratando de detectar dónde están los ojos en una cara, una característica es que los ojos son más oscuros que las mejillas. En ese caso, se podría aplicar una región rectangular como la mostrada a la izquierda para detectar ojos. Así pues, una característica de Haar (o Haar-like features) común para la detección de caras es un conjunto de dos rectángulos adyacentes, uno debajo de otro, que coincidirán con la región de los ojos y mejillas. De esta forma se definen una serie de características que permiten detectar de una forma rápida y con una gran probabilidad de acierto lo que es y lo que no es una cara.

Una vez comentadas las características de Haar, el algoritmo de detección en sí tiene dos etapas. Primero una etapa de entrenamiento, en la que a una serie de filtros en cascada se les pasan unos patrones positivos (que coinciden con el objeto buscado) y negativos (no tienen el objeto) de forma que el sistema aprenda y se forme un modelo de las características de Haar del objeto a detectar. Cada una de estas características por separado (llamadas características débiles) no son suficientes para determinar si la imagen presentada contiene o no el objeto buscado. Pero una cascada de todas las características que debe cumplir el objeto (hay indicios de detección de ojos, indicios de detección de boca, indicios de detección de nariz) sí son suficientes para poder discernir si la imagen o porción de imagen se corresponde o no con el objeto buscado, en este caso una cara.

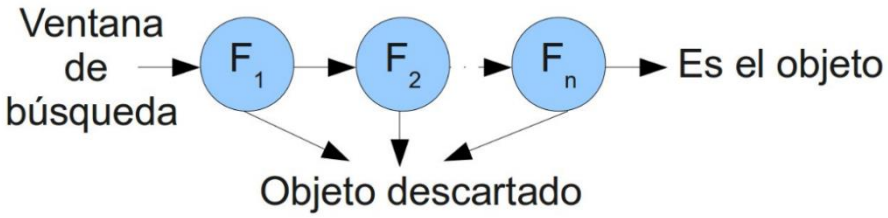


Figura 5: Cascada Haar

La ventaja de este tipo de detectores es que las características de Haar son rápidas de obtener, y sólo se calculan todas las características en el caso de que la región de la imagen en cuestión pase todos los clasificadores débiles. En el momento en que la imagen no pasa el umbral de uno de los clasificadores débiles se descarta, ahorrándose el cálculo del resto de clasificadores.

2.4.2. Local Binary Patterns

En el trabajo presentado por He y Wang en [17] se describe por primera vez una nueva técnica para el análisis de texturas en imágenes a través de las llamadas unidades de textura. Esta técnica consiste en evaluar la disposición espacial y el contraste de los píxeles en pequeñas regiones de la imagen. Como resultado se obtiene un espectro de textura por cada región que permite identificar el tipo de textura global de la imagen. Desafortunadamente esta técnica no tuvo ninguna relevancia ni teórica ni práctica, al menos en sus inicios. Unos años más tarde fue cuando [21] utilizaron la idea de las unidades de textura, pero dándoles una nueva visión. En este caso las unidades de textura son codificadas mediante un número binario de tal manera que cada tipo de textura pueda ser identificado fácilmente. A este nuevo método lo llamaron características LBP. El operador LBP (Local Binary Patterns) funciona de la siguiente manera. Sobre entornos de la imagen de 3x3 píxeles, se compara la intensidad de cada uno de los ocho píxeles vecinos con la intensidad del píxel central. Si la intensidad del píxel vecino es mayor o igual que la del píxel central, se le asigna a la posición del píxel vecino un valor de 1 o, en caso contrario, un valor de 0. Una vez comparados todos los píxeles, se tendrá un conjunto de ceros y unos que representarán un número binario. Multiplicando cada posición del número binario por su peso en decimal y sumando todo, se obtiene el valor de la característica LBP con el que se etiqueta el píxel central. En la Figura 6 se muestra un ejemplo para comprender mejor este proceso:

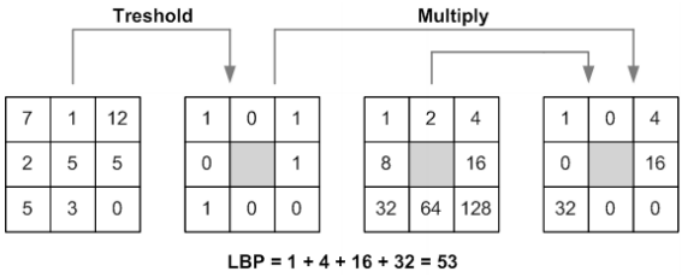


Figura 6: TLB

Una descripción más formal del operador LBP se obtiene mediante la expresión (1), con la posición del pixel central , el número de pixeles de la vecindad, la intensidad de los pixeles vecinos, la intensidad del pixel central y la función (2).

$$LBP(X_{c,}Y_c) = \sum_{p=0}^{p-1} s(I_p - I_c)2^p \quad (1)$$

$$s(x) = \begin{cases} 1, x \geq 0 \\ 0, si\ no \end{cases} \quad (2)$$

En definitiva, para cada píxel de la imagen se obtiene un valor asociado que indica el tipo de textura que presentan sus 8 píxeles vecinos. Con esta combinación de valores, a lo máximo se

podrán representar 28 o 256 tipos de unidades de texturas. Así pues, calculando el histograma de la característica LBP y analizando sus valores, se puede extraer un patrón identificativo de texturas que permita clasificar la imagen.

Este mismo procedimiento puede ser aplicado para la detección de objetos. Si se conociese el patrón de texturas que presenta el objeto, sería tan fácil como calcular la característica LBP de la imagen donde se está buscando, obtener su histograma y encontrar su correspondencia por comparación.

2.4.3. Histograma de Gradientes Orientados

El procedimiento de HOG (Hisogram of oriented gradients) fue propuesto por Dalal y Triggs [9] como un método para la detección de objetos. No obstante, el ámbito donde más destaca su uso, ya indicado en el propio artículo en el que se describió, es en la detección de personas en imágenes estáticas. Lo usaban como característica principal para su detector de personas con el que conseguían tasas de acierto cercanas al 100% y con una tasa de falsos positivos por ventana cercana al 40%. Estos resultados se obtuvieron para una base de datos de imágenes en las que aparecían personas sin oclusiones, o con muy poca oclusión.

El proceso se basa en la división de la imagen en ventanas de acuerdo con una configuración en rejilla. Para cada una de estas ventanas se obtiene un histograma cuyas entradas se corresponden a las distintas orientaciones de los gradientes. El proceso a grandes rasgos de obtención del HOG para una ventana dada es el siguiente:

1. Cálculo de los gradientes de la ventana.
2. Discretización mediante histograma de los gradientes calculados.

Para el cálculo de gradientes de la ventana se aplican los dos filtros unidimensionales siguientes: [–1 0 1] y [–1 0 1]’. Esto da lugar a gradientes con signo, que serán convertidos a sin signo en el paso siguiente.

Como se explica en [9], la dirección del gradiente no aporta información significativa, por lo que únicamente se tiene en cuenta la orientación de éste. La rejilla que sitúan sobre la imagen tiene un tamaño de celda óptimo de 6x6 píxeles. Los histogramas de orientaciones se crean acumulando un voto por cada posible orientación, previamente discretizada a un cierto rango. Este voto tiene como valor la magnitud de la orientación ya que aunque se probó con variantes (el cuadrado de la magnitud, su raíz cuadrada, etc.) la magnitud por sí misma es la que mejores resultados proporciona.

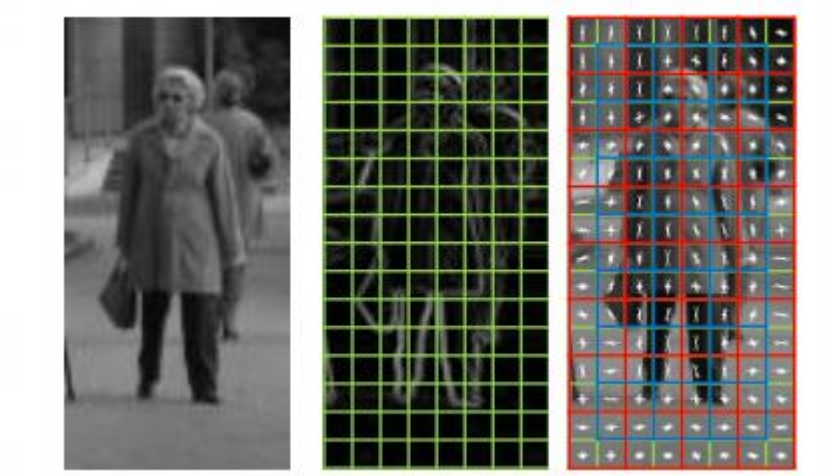


Figura 7: Fotografía de un peatón (izquierda) y su descriptor HOG (derecha) [3]

Tras obtener los descriptores se aplica un clasificador lineal basado en el procedimiento de aprendizaje conocido como SVM (Máquina de vector de soporte) para el entrenamiento y posterior detección de la frontera que determina el objeto.

2.5. Aprendizaje Automático

El aprendizaje automático o aprendizaje de máquinas es una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. Es, por lo tanto, un proceso inductivo o deductivo de extracción del conocimiento. El aprendizaje automático y las técnicas estadísticas son dos campos de actuación que comparten en común el tratamiento masivo de datos, más explícitamente se encuadran en lo que se conocen como técnicas de “big-data” o minería de datos, ya que las dos disciplinas se basan en el análisis de datos. Sin embargo, el aprendizaje automático se centra más en la estimación de parámetros teniendo en cuenta la complejidad computacional de los problemas. En este sentido, muchos problemas son de clase NP-hard, por lo que gran parte de la investigación realizada en aprendizaje automático está enfocada al diseño de soluciones factibles a esos problemas. En términos muy generales, el aprendizaje automático puede ser visto como un intento de automatizar algunas partes del método científico mediante métodos matemáticos.

El aprendizaje automático constituye un paradigma de la Inteligencia Artificial aplicado para la resolución de múltiples problemas, entre los que se incluyen motores de búsqueda, diagnósticos médicos, detección de fraude en el uso de tarjetas de crédito, análisis del mercado de valores, clasificación de secuencias de ADN, reconocimiento del habla y del lenguaje escrito, juegos,

robótica y por supuesto reconocimiento e identificación de caracteres como es el caso que nos ocupa.

2.6. Aproximaciones similares

Un caso de estudio similar al problema que recoge esta memoria, es el reconocimiento automático de matrículas de vehículos, conocido por sus siglas en inglés ANPR (Automatic Number Plate Recognition).

En [5] y [25], podemos observar cómo el proceso de reconocimiento consta de los siguientes seis procedimientos [11]:

1. Localización de la matrícula - responsable de encontrar y aislar la matrícula en la imagen.
2. Orientación y tamaño de la matrícula - compensa los ángulos que hacen que la matrícula aparezca inclinada respecto de la horizontal y ajusta las dimensiones al tamaño requerido.
3. Normalización - ajusta el brillo y el contraste de la imagen.
4. Segmentación de los caracteres - encuentra los distintos caracteres presentes en la matrícula.
5. Reconocimiento óptico de caracteres.
6. Análisis sintáctico y geométrico - comprueba los caracteres encontrados y sus posiciones con las reglas específicas del país al que pertenece la matrícula.

Figura 8: Detección de la ROI (izquierda), procesado y segmentación (derecha)

En el proceso de investigación previo del proyecto, se han encontrado diferentes soluciones para el reconocimiento aplicado a contadores de suministros como luz, agua, incluso específicamente de gas.

De [33], obtenemos la siguiente Figura 9. Para la detección de la ROI (Región de Interés), se utiliza un sistema de redes neuronales basadas en el algoritmo MNOD. En [20] y [26] encontramos dos resoluciones similares.

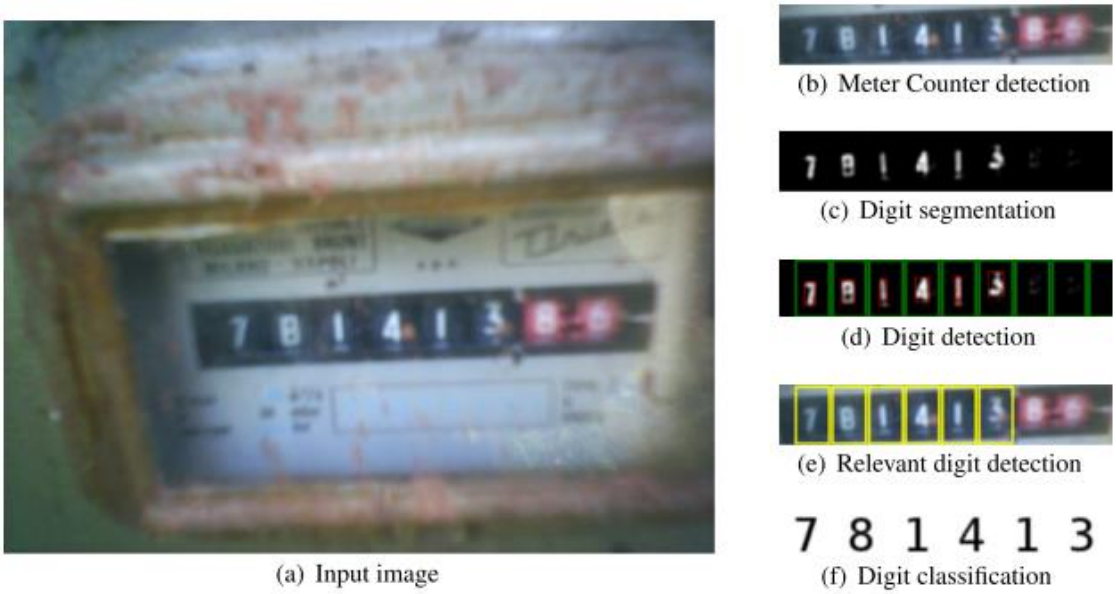


Figura 9: Fases del reconocimiento [39]

Por último la aproximación más cercana a nuestro proyecto la encontramos en la publicación [6], que surge como encargo de la empresa energética francesa GDF Suez. El esquema de resolución planteado por Choutien y Schaeffer lo podemos observar en la Figura 10.

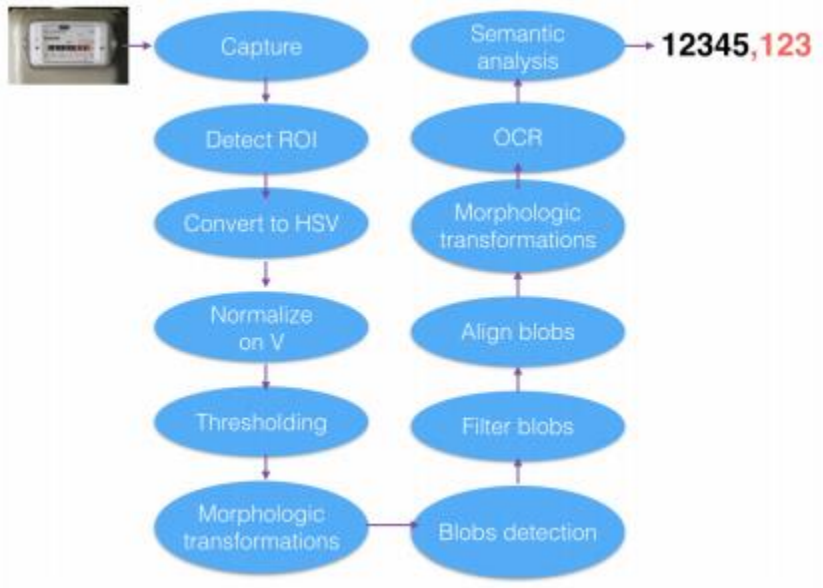


Figura 10: Etapas de la solución [6]

Para mejorar la efectividad del algoritmo, introducen información basada en el contexto, de modo que se tienen en cuenta las lecturas anteriores para reducir los casos posibles (la lectura anterior siempre refleja menos metros cúbicos que la posterior). Esto soluciona algunos casos conflictivos, como en los que la rueda del dígito está situada entre dos números, como podemos ver en la siguiente Figura 11. Dicha implementación está siendo utilizada actualmente en la aplicación de la empresa GDF Suez en iOS y Android.

Figura 11: Caso conflictivo en los últimos dígitos [6]

3. Técnicas y metodologías aplicadas

Para la obtención de los resultados finales relativos al tratamiento de imágenes hemos aplicado distintas técnicas y metodologías en diversos ámbitos. Para la consecución de una imagen más clara, imagen binaria, de los dígitos del consumo del contador, hemos aplicado varias operaciones de tratamiento de imágenes, exactamente son las siguientes: detección de bordes y binarización de imágenes. El objetivo de estas técnicas de pre-procesado estriba en la necesidad de obtener una imagen con la máxima calidad posible para garantizar que las técnicas de OCR, utilizadas para el reconocimiento de caracteres numéricos obtengan el mayor éxito posible. Ello se consigue gracias a que dichas técnicas están encaminadas a la eliminación de ruido en las zonas de interés de las imágenes.

El código de barras existente en todos los contadores identifica de forma unívoca al usuario y constituye un elemento esencial para determinar por un lado la ubicación del código a identificar y por otro el propio código. Se basa en un sistema de representación numérica que una vez decodificado permite la identificación del código por este procedimiento, siendo un sistema de comprobación con respecto a los caracteres numéricos identificados.

3.1. Análisis de imágenes

Las imágenes digitales básicas en color están formadas por tres componentes espectrales correspondientes a cada uno de los canales R (rojo), G (verde) y B (azul) en el modelo de color RGB (Red, Green, Blue). Cada uno de ellos se representa mediante una matriz cuyos elementos, en una posición determinada (x,y), son los píxeles tomando valores numéricos en un rango determinado, normalmente entre [0, 255] cuando la representación de cada píxel es de 8 bits. Estas imágenes en color se pueden transformar convenientemente a diferentes modelos de color entre los que se encuentra el modelo HSV, también con tres componentes, donde la componente V es la de intensidad o imagen de gris, en este caso se trata de una única matriz de gris con valores en el rango [0,255] con la misma representación anterior.

Sobre la imagen de intensidad resulta habitual realizar determinado tipo de operaciones o transformaciones que generan imágenes con dos únicos valores 0 ó 255, es decir valores negro o blanco respectivamente, que desde el punto de vista lógico se corresponden con valores 0 y 1. Estas imágenes se conocen como imágenes binarias que se obtienen aplicando técnicas conocidas como de binarización, basadas en la aplicación de un umbral, cuyas técnicas se conocen como técnicas de umbralización o *thresholding*.

3.1.1. Umbralización

Como se ha mencionado previamente la técnica de umbralización constituye uno de los métodos de segmentación más simples que existe, la manera en la que se aplica es muy sencilla, se elige un valor de umbral, de forma fija o dinámica según la zona. Considerando dicho umbral, se reemplaza cada valor de intensidad en la imagen de gris mayor que él por 1 (blanco), o por 0 (negro) si es menor que el umbral. La binarización de una imagen en la aplicación que se propone tiene como objetivo separar la imagen en varios segmentos, normalmente para diferenciar los objetos de una imagen del fondo. Esto es precisamente la técnica a aplicar en el reconocimiento de las tiras de caracteres numéricas a identificar en los contadores.

Existen distintos métodos de umbralización, algunos de ellos automáticos, que se usan según el tipo de imágenes que se quiera obtener. Las distintas modalidades son:

- *Binaria*: los valores por encima del umbral se ponen a 255 y el resto a 0. Existe también la binaria invertida cuyo proceso resulta ser justo el contrario, los valores mayores se ponen a 0 y los menores a 255.

- *Truncada*: los valores de la matriz por encima del valor umbral se ponen a 255 y por debajo se mantienen.

- *A cero*: los píxeles por encima del umbral mantienen su valor y los menores se ponen a 0. También existe la considerada a cero invertida, en la que los valores superiores al umbral son 0 mientras que los inferiores se conservan.

En la imagen de la Figura 12 se muestran ejemplos ilustrativos sobre las técnicas de binarización expresadas previamente.

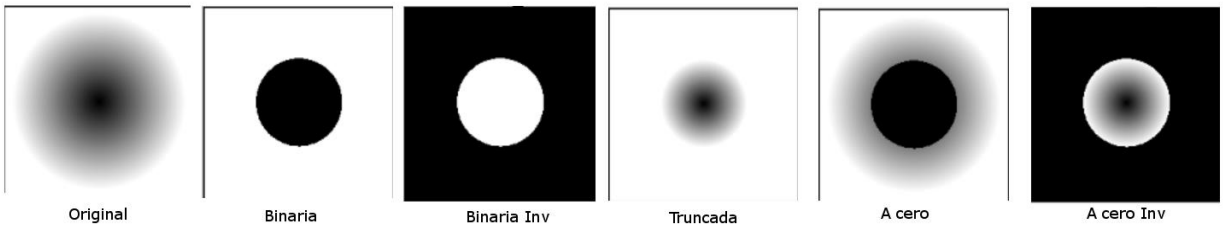


Figura 12: Resultados segmentación con umbral fijo

En las imágenes mostradas en la Figura 12, se observan perfectamente los resultados de la binarización. En la Figura 13 se muestran igualmente ejemplos de binarización en el caso de las imágenes reales de los contadores objeto del procesamiento.

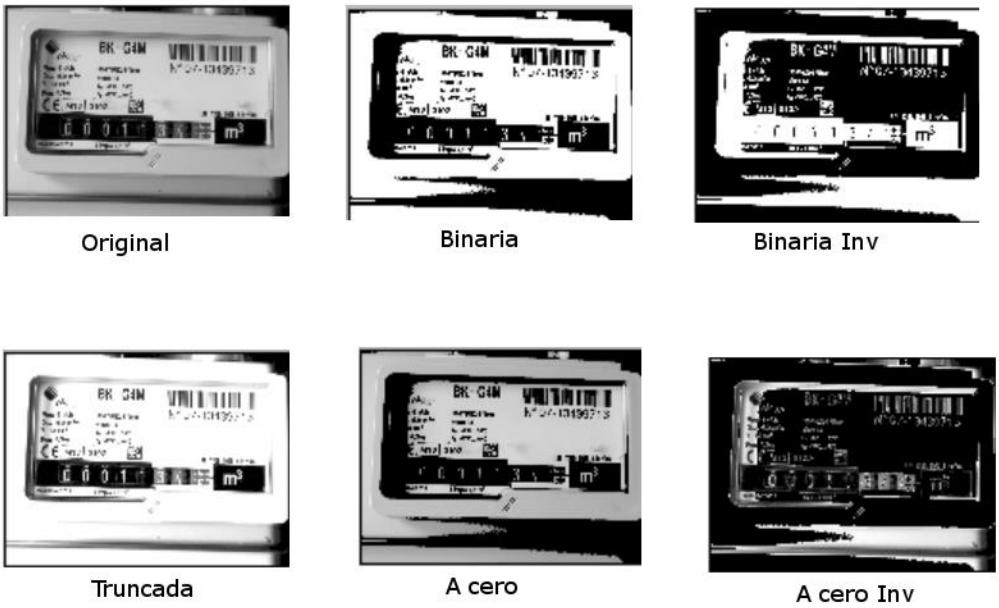


Figura 13: Resultados segmentación con contadores reales con umbral fijo

Se puede observar en las imágenes de la Figura 13 que la zona de interés, que es donde se encuentra los dígitos del contador, resulta difícilmente reconocible por un OCR. Ya que el valor de umbral fijado de forma manual no resulta factible. Debido a ello es necesario considerar la aplicación de técnicas de umbralización adaptativas, de forma que el valor de umbral varía según la zona de la imagen a tratar, adaptándose al entorno de esa zona.

Los dos tipos de umbralización adaptativa utilizados en la aplicación que se presenta son:

- *Mediana:* el valor de umbral se calcula computando la mediana del área de vecindad considerada alrededor de cada píxel. El valor de umbral se calcula para todos los píxeles de la imagen.

- *Media:* como en el caso anterior se considera una ventana sobre la que se obtiene el valor medio, que determina el umbral en la ventana. .

Las imágenes en la Figura 14 muestran dos ejemplos de imágenes binarias obtenidas mediante los métodos mencionados.

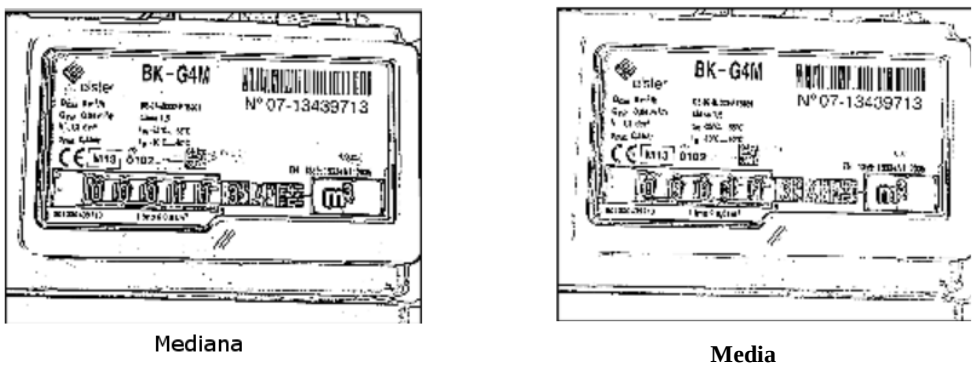


Figura 14: Imágenes binarizadas de contadores reales con umbral dinámico

A partir de los resultados mostrados en la Figura 14, se puede comprobar que con los métodos adaptativos, las zonas de interés para aplicar un OCR resultan mucho más nítidas que las obtenidas con los métodos de umbralización fijos.

3.1.2. Suavizado

Uno de los problemas más comunes que nos encontramos a la hora de analizar imágenes digitales es el ruido u otros efectos no deseados, producidos por perturbaciones a la hora de realizar la captura. Con el suavizado lo que intentamos conseguir es mejorar la calidad de la imagen, al eliminar píxeles no deseados en la imagen. En tratamiento de imágenes son típicas las cuatro siguientes operaciones de suavizado:

- Media del entorno de vecindad
- Gaussiano
- Mediana del entorno de vecindad
- Bilateral

3.1.2.1 Suavizado mediante la media del entorno de vecindad

Dada una imagen se obtiene otra imagen suavizada, cuya intensidad para cada píxel es la media de un entorno de vecindad predefinido, que puede verse reflejado en la fórmula (3), donde M y N son las dimensiones del entorno de vecindad considerado.

$$k = \frac{1}{M \times N}\begin{bmatrix} 1 & 1 & 1\ldots & 1 & 1 \\ 1 & 1 & 1\ldots & 1 & 1 \\ 1 & 1 & 1\ldots & 1 & 1 \\ . & . & \ldots. & . & . \\ 1 & 1 & 1\ldots & 1 & 1 \end{bmatrix} \quad (3)$$

Si el entorno de vecindad es una ventana de dimensión 3x3 y tenemos la siguiente matriz el resultado sería el que se muestra a continuación. Puede verse en el siguiente ejemplo en el que el píxel considerado de ruido con valor 1 adquiere un valor de 7, que es más próximo al resto de valores, en este caso 8, logrando así un suavizado.

Imagen original

$$\begin{bmatrix} 8 & 8 & 8 & 8 & 8 \\ 8 & 8 & 1 & 8 & 8 \\ 8 & 8 & 8 & 8 & 8 \\ 8 & 8 & 8 & 8 & 8 \\ 8 & 8 & 8 & 8 & 8 \end{bmatrix} \quad k = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Imagen resultante

$$\begin{bmatrix} 8 & 8 & 8 & 8 & 8 \\ 8 & 8 & 7 & 8 & 8 \\ 8 & 8 & 8 & 8 & 8 \\ 8 & 8 & 8 & 8 & 8 \\ 8 & 8 & 8 & 8 & 8 \end{bmatrix}$$

3.1.3. Extracción de bordes: Sobel

El operador Sobel es un operador primera derivada que se usa para la detección de bordes de una imagen. Se aplica en cada punto o píxel de la imagen, para determinar el cambio de intensidad en ese píxel. Mediante este operador se determina tanto la magnitud como la dirección del cambio. Si el cambio es muy pronunciado es probable que el punto analizado sea un borde; por el contrario, si no se trata de un cambio brusco, no será considerado como punto de borde.

Matemáticamente, se trata de aplicar la primera derivada para calcular dos componentes, una horizontal y otra vertical mediante sendos núcleos, generalmente de dimensión 3x3, uno para los cambios horizontales y otro para los verticales (4). Cada núcleo se aplica para todos los píxeles de la imagen, con lo cual, en un punto dado definido por la vecindad A se obtienen las componente Gx y Gy que constituyen las componentes del vector gradiente G = [Gx, Gy], cuyo módulo, determina la magnitud del gradiente, que puede aproximarse como |G| = |Gx| + |Gy| (5).

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \times A \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \times A \tag{4}$$

$$G = \sqrt{{G_x}^2 + {G_y}^2} \tag{5}$$

En la Figura 15 se muestra una imagen original, así como sus correspondientes magnitudes |Gx| y |Gy|, así como el módulo |G|. En estas imágenes se puede observar claramente los bordes y las diferencias entre la que contiene los bordes horizontales y la que tiene los bordes verticales. En la imagen que representa el módulo del gradiente se aprecia cómo se obtienen mejor los bordes en las zonas más claras de la imagen, zona inferior, que en las más oscuras, zona superior. En esta imagen se aprecian claramente los bordes.

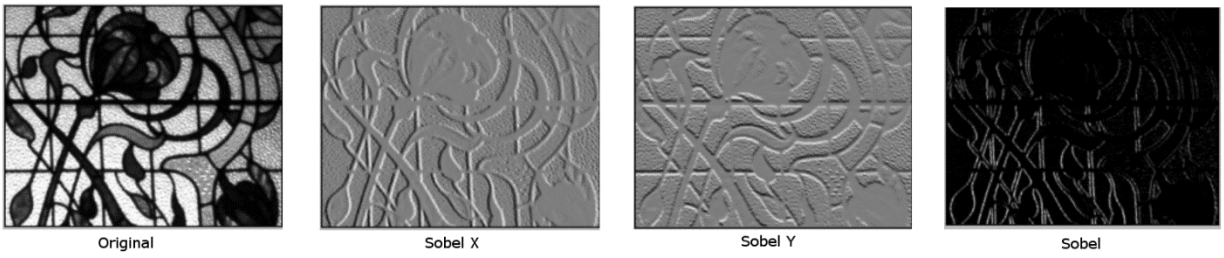


Figura 15: Resultados Sobel 1

Mediante la aplicación de los operadores de Sobel a las imágenes reales de los contadores utilizados en este proyecto, se obtienen los resultados que se muestran en la Figura 16. En estas imágenes se puede observar claramente los bordes y las diferencias entre la que contiene los bordes horizontales y la que tiene los bordes verticales. La última imagen representa el módulo

del gradiente en la que se aprecia que se obtienen mejor los bordes en las zonas más claras de la imagen, zona inferior, que en las más oscuras, zona superior. En esta imagen se aprecian claramente los bordes.

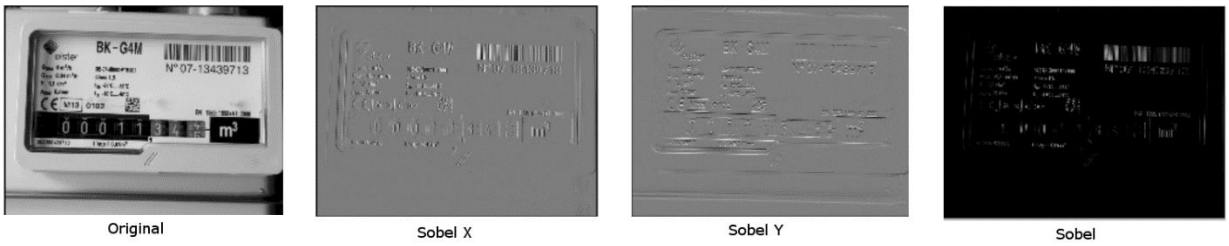


Figura 16: Resultados Sobel 2

En general, en las imágenes de contadores utilizadas en el presente proyecto, aunque hay zonas donde sí se distinguen los bordes, éstas no son realmente zonas de interés para la lectura del contador, esto impide la utilización eficiente de los operadores de bordes para detectar las zonas de interés para las lecturas pretendidas en los contadores. Debido a esto hay que aplicar distintas técnicas para poder obtener la zona lo más nítida posible.

3.2. Análisis morfológico de una imagen

Existen cuatro principios aplicables al conjunto de transformaciones morfológicas aplicables a las imágenes de los contadores [23], son los siguientes:

- Compatibilidad con la translación
- Compatibilidad con el cambio de escala
- Conocimiento local
- Continuidad-semi-superior

El objetivo de las transformaciones morfológicas es la delimitación de figuras geométricas a través de un objeto conocido, llamado elemento estructural. Las cuatro transformaciones más simples son: dilatación, erosión, apertura y cierre. Vamos a explicar las dos primeras más detalladamente que son las que hemos usado para alcanzar nuestros objetivos. La morfología se puede usar por ejemplo, para unir zonas que se han separado durante la segmentación o para separar zonas que se hayan unido

3.2.1. Creación del elemento estructural

El elemento estructural realmente es otra matriz que va recorriendo todos los píxeles de la imagen inicial. La creación de esa matriz varía dependiendo del objeto o área de la imagen que se quiere procesar. Normalmente los elementos estructurales suelen ser formas geométricas como rectángulos, círculos o polígonos regulares.

Figura 17: Elementos Estructurales

3.2.2. Erosión

Con la erosión lo que se busca es comprobar si hay objetos que están contenidos en el elemento estructural, es decir, si el objeto es más pequeño que el elemento, no aparecerá en la imagen resultante, en cambio si el objeto es mayor o igual sí que aparecerá.

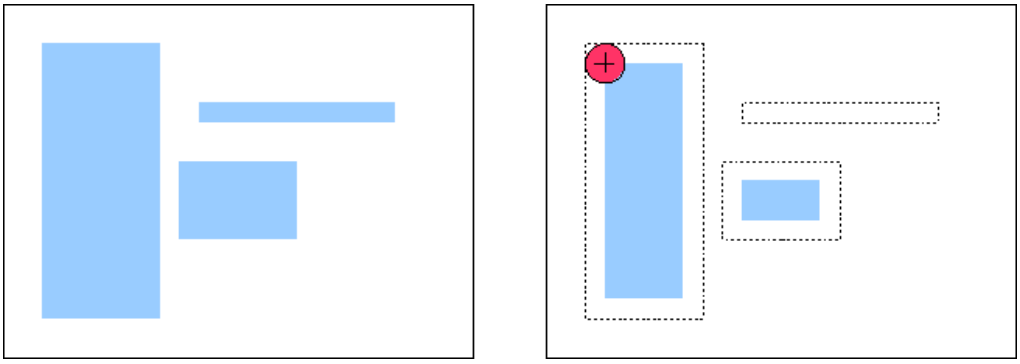


Figura 18: Erosión

Se aprecia claramente que la imagen resultante, que es la imagen original menos las zonas determinadas por el elemento estructural, no pertenece completamente. Por eso la barra superior derecha al ser más delgada que el círculo, desaparece en la imagen final, mientras que los otros dos rectángulos dado su tamaño superior al elemento estructural permanecen.

Con un ejemplo de dos matrices se observa más claramente el proceso de erosión.

Imagen original

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Elemento Estructural

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Imagen resultante

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

La matriz original tiene la zona interna blanca y al recorrerla con la matriz del elemento estructural, la imagen resultante sólo tiene los píxeles a 1 cuando en un píxel de la imagen original coincide con todos los píxeles del elemento estructural.

3.2.3. Dilatación

La dilatación es un proceso dual a la erosión. Lo que se busca es que con que haya un píxel que se encuentre en el elemento estructural, se expandirá la imagen original. Con esto conseguimos que los objetos con un tamaño igual al del elemento estructural no se modifiquen pero que los bordes de dicho objeto se amplíen.

Imagen original Elemento estructura Imagen resultante

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Como vemos todos los píxeles que son vecinos del objeto son transformados además de los que forman parte del objeto. Por eso la dilatación es una expansión.

3.3. Principios matemáticos del código de barras

La primera patente de código de barras fue registrada en octubre de 1952 [38] por los inventores Joseph Woodland, Jordin Johanson y Bernard Silver en Estados Unidos. La implementación fue posible gracias al trabajo de los ingenieros Raymond Alexander y Frank Stietz. El resultado de su trabajo fue un método para identificar los vagones del ferrocarril utilizando un sistema automático. Sin embargo, no fue hasta 1966 que el código de barras comenzó a utilizarse comercialmente y no tuvo un éxito comercial hasta 1980.

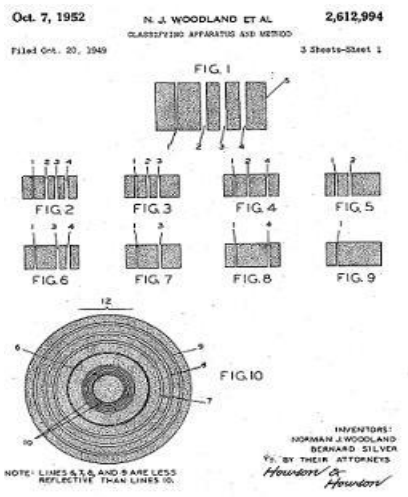


Figura 19: Fragmento de la patente US 2612994 A [38]

El estándar europeo es conocido como EAN y su implementación más conocida es el EAN13 (13 dígitos).

La composición es la siguiente:

- Código del país donde radica la empresa, compuesto por 2 dígitos.
- Código de empresa. Es un número compuesto de 5 dígitos, que identifica al propietario de la marca. Es asignado por la asociación de fabricantes y distribuidores.
- Código de producto. Completa todos los dígitos restantes, menos el de control.
- Dígito de control. Para comprobar el dígito de control (por ejemplo, inmediatamente después de leer un código de barras mediante un escáner), se numeran los dígitos de derecha a izquierda. A continuación se suman los dígitos de las posiciones impares, el resultado se multiplica por 3, y se le suman los dígitos de las posiciones pares. Se busca la decena inmediatamente superior y se le resta el resultado obtenido. El resultado final es el dígito de control. Si el resultado es múltiplo de 10 el dígito de control será 0.

En el siguiente ejemplo podemos ver el procedimiento de cálculo del dígito de control para la Figura 20:

Figura 20: Estructura EAN 13

1) Suma los dígitos en las posiciones impares (comenzando de derecha a izquierda): 8 + 7 + 1 + 5 + 0 + 4 = 25
2) Multiplica el resultado en el paso uno por 3: 25 x 3 = 75
3) Suma los dígitos en las posiciones pares (salvo el dígito No. 12): 4 + 0 + 0 + 1 + 8 + 8 = 21
4) Suma el resultado del paso tres con el del paso dos: 75 + 21 = 96
5) Se realiza el módulo del múltiplo de 10 más cercano y nos da como resultado el dígito de control que es **4**.

3.4. OCR

Tras la revisión histórica sobre el desarrollo y progreso del OCR realizado en el punto 2.3, podemos deducir la enorme cantidad de implementaciones existentes, si bien casi todas siguen el esquema de la Figura 21, a su vez tomada como ejemplo de [14].

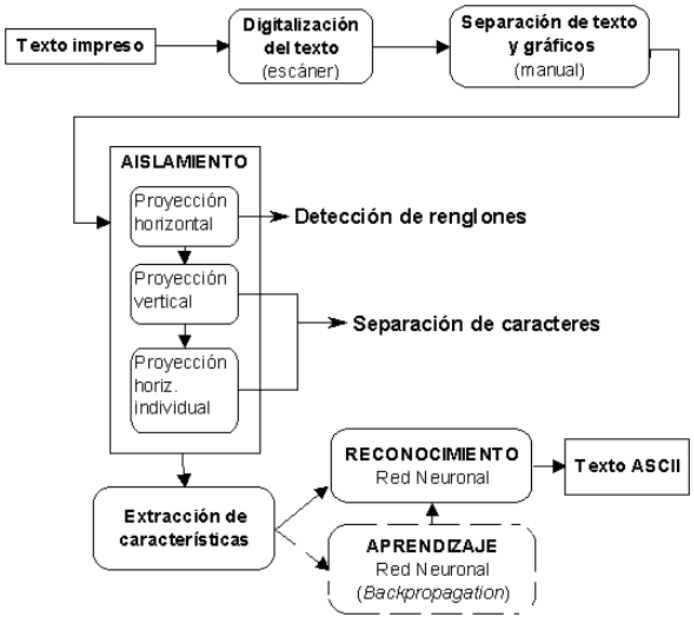


Figura 21: Esquema de reconocimiento de caracteres [14]

En la Figura 22, podemos observar una comparativa de los diferentes motores de OCR más utilizados aplicados a la región de interés de la lectura del consumo en contadores de gas y su respectiva salida. En este punto nos centraremos en la implementación de Tesseract, que es el motor utilizado en el proyecto, los motivos de la elección se exponen más adelante en el punto 4.

Test 6: Number OCR

OCR Service	Result	Output
Abbyy Cloud SDK	Good	0491024,3
Google Docs OCR	Fail	(no output)
OnlineOCR	Fail	0 6 4 0
i2 OCR	Fail	01} § 7 6,2 F.
FreeOnlineOCR	Fail	(no output)
Tesseract Online	Fail	f11l§7fi,2Z?3§-

Figura 22: Comparativa de diferentes motores de OCR para lectura de contadores [1]

Tesseract OCR surgió como un desarrollo del departamento de experiencias de usuario de HP desde el año 1985, poniéndose entre la lista de los tres primeros respecto a la tasa de acierto en la competición anual de la Universidad de Las Vegas del 1995. En el año 2005 HP liberó el proyecto y lo licenció como código abierto bajo una licencia Apache. Desde el año 2006 está impulsado por Google.

En [28], publicado por Google encontramos una introducción de las diferentes fases del proceso de reconocimiento, como podemos ver en la Figura 23.

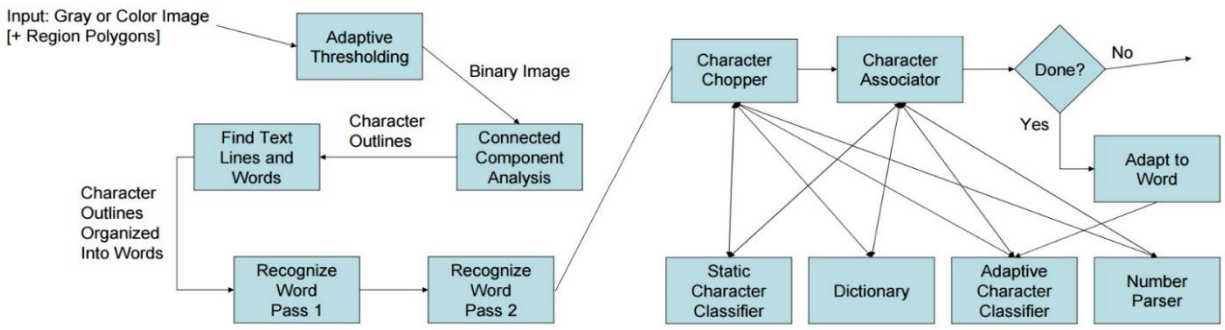


Figura 23: Proceso de Tesseract OCR [29]

Actualmente está entrenado para reconocer caracteres de seis idiomas y todos los caracteres del formato de codificación UTF-8. El clasificador o diccionario puede reentrenarse con herramientas como jTextBoxEditor [30].

4. Fases preliminares de desarrollo

Tras la propuesta inicial, se realizó un estudio de los requisitos del proyecto junto a las posibles soluciones. Hay que diferenciar en dos apartados las fases preliminares. El primer apartado trata sobre la lógica de la aplicación, el framework utilizado, la persistencia de la información y la experiencia de usuario. El segundo se centra en el tratamiento de la imagen y la lectura de los dígitos.

En cuanto al primer apartado, éstos fueron los puntos propuestos en el esquema planteado:

- Sistema de registro y login.
- Gestor de archivos para subir las imágenes.
- Base de datos para guardar los datos del registro y las propias imágenes.
- Elección entre aplicación de escritorio o web.
- Interfaz sencillo y amigable que permitiese la corrección de las lecturas en caso de fallar el reconocimiento en algún digito.
- Todo ello, bajo la necesidad de que fuera un sistema portable y multiplataforma.

Respecto al segundo apartado, se identificaron los siguientes planteamientos sobre los que deberíamos encontrar la solución:

- Selección de un lenguaje y un framework para el tratamiento de las imágenes.
- Lectura del número de serie.
- Detección de la región de interés de los dígitos del contador.
- Posibilidad de leer otro tipo de contadores.
- El reconocimiento debe ser válido en entornos reales, la imagen puede contener otros elementos distintos al contador.
- Reconocimiento de caracteres

Para todos los puntos planteados, que necesitan una solución, se ha realizado un proceso de investigación, teniendo como norma siempre el uso de software libre o en el caso de no encontrarse alternativas libres, software de código abierto.

Por lo tanto, la primera decisión necesaria, era la elección de código de programación, que está fuertemente ligada a la elección del framework y de la librería de procesamiento de imágenes. Las dos alternativas respecto al procesamiento de imágenes eran Matlab y OpenCV. El problema de Matlab (y su alternativa libre Octave) es que la integración con otro software y lenguajes resulta un tanto complicado. Por otro lado, OpenCV es la Liberia de Visión por Computador de mayor uso de la comunidad, mejor documentada y puede ser utilizada desde casi todos los lenguajes de programación existente.

Una vez elegida OpenCV como librería de tratamiento de imágenes, los lenguajes de programación más documentados para esta librería son C++/C# y Python. A pesar de que OpenCV está escrita en C++, lo que conlleva un mayor rendimiento en términos

computacionales, y de la comparativa que podemos ver en la Figura 24 respecto del rendimiento entre varios lenguajes, donde C++ obtiene los mejores resultados, si bien, el lenguaje es muy dependiente tanto de la arquitectura como del sistema operativo, lo que nos impide una fácil portabilidad del software. En cualquier caso, cómo podemos ver en la Figura 25, el wrapper de Python para OpenCV ha resultado ser más efectivo a lo largo de las actualizaciones. Esta figura se ha extraido de [32], donde se describen las ventajas del uso de Python en aplicaciones de visión por computador, para aplicaciones que no necesiten de soporte de funcionalidades en tiempo-real y no residan en dispositivos empotrados.

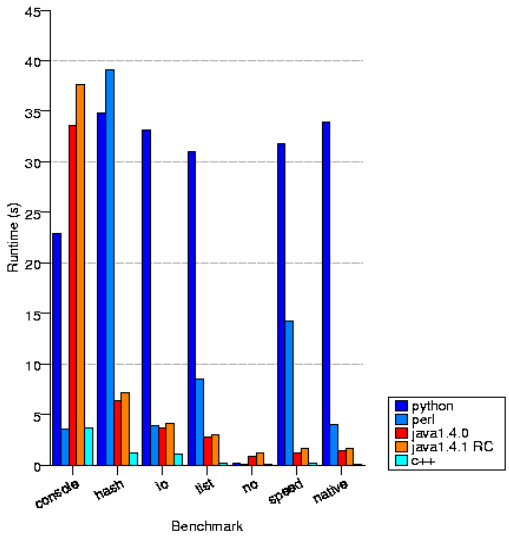


Figura 24: Comparativa de tiempos en diferentes lenguajes

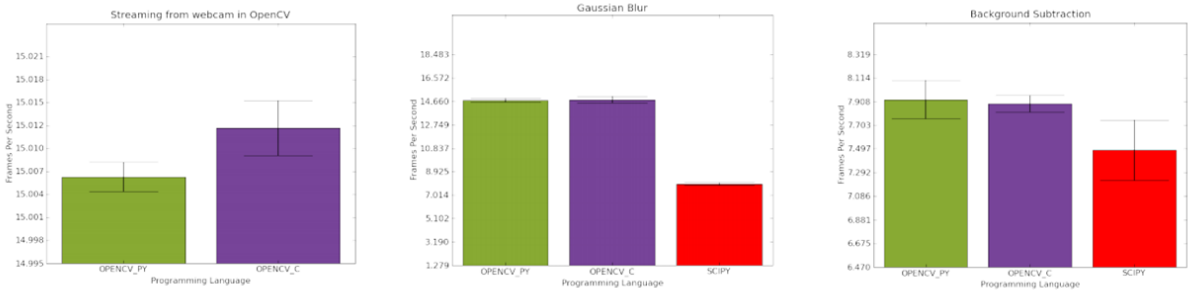


Figura 25: Comparativa de rendimiento de funciones de OpenCV

Con ambas decisiones tomadas, pasamos a analizar las ventajas y desventajas de las aplicaciones de escritorio y de las aplicaciones web. Para las primeras demostraciones, se utilizó el paquete tkInter para diseñar la interfaz gráfica. Durante la etapa de desarrollo era una opción válida, pero no resultaba óptima para las opciones de futuro que permitiera hacer uso del backend tanto desde un ordenador, como desde un smartphone. Por lo que planteamos un diseño de arquitectura, similar al Modelo-Vista-Controlador, que permitiese mantener un modelo y

controlador, con independencia de la vista. Al haber elegido Python como lenguaje para el backend, buscamos información del framework Django que encajaba con nuestra arquitectura planteada y además permite el desarrollo de la aplicación web, el sistema de control de usuarios, el uso en un futuro de llamadas REST desde una aplicación de smartphone y es compatible con diferentes motores de base de datos, tanto SQL (Structured Query Language) como NOSQL. Esta elección nos permite el desarrollo de la interfaz de usuario haciendo uso de HTML (HyperText Markup Language) y CSS (Cascading Syle Sheets), cuyo tiempo de desarrollo es menor en comparación con otras tecnologías.

Para obtener las diferentes regiones de interés, a partir de aquí llamadas ROI, comenzamos con la búsqueda de la zona roja del contador, tal y como se puede ver en la Figura 26, y a través de esa región, realizar un desplazamiento de la ventana lateral hacia la izquierda para obtener la región de los dígitos de la lectura, al mismo tiempo que una ventana de desplazamiento hacia arriba para localizar el número de serie.

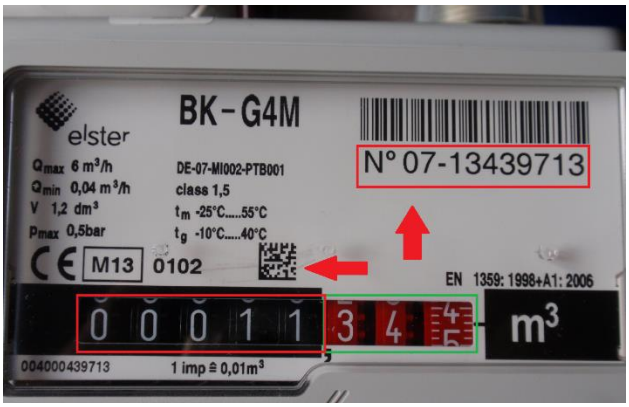


Figura 26: Detección de la región roja del contador

El problema es que en las imágenes de entornos reales, en las que la imagen no sólo encuadra el contador, sino que también contiene los elementos de la pared, tubería de suministro de gas y otros elementos, es fácil encontrar falsos positivos de la componente roja en elementos tales como el óxido y el cobre de la tubería. Por lo que esta solución se desestimó.

Para obtener el número de serie del contador, se debería primero obtener la ROI, y luego pasar un sistema de OCR para interpretarlo. Una forma de detectar la ROI, es utilizar el código de barras que se encuentra justo encima del número de serie, que puede ser detectada mediante operaciones morfológicas como se describe en [18]. Tras implementar una prueba de concepto de la detección, al escanear el código de barras comprobamos que el número estaba reflejado también en el código de barras por lo que ya no era necesario aplicar OCR, ya que la lectura del código de barras tiene una tasa de acierto superior al propio OCR.

Respecto a la región de interés, tras desechar la opción de la detección de la zona roja, se probó la búsqueda de la región mediante los contornos, obteniendo buenos resultados como observamos en la Figura 27.

Figura 27: Detección de la ROI haciendo uso de deteccion de contornos

El problema de esta solución es que la tasa de acierto en la detección no era suficiente en imágenes en las que la fotografía encuadrase más elementos aparte del contador. Como ambos integrantes del proyecto hemos cursado la asignatura de Aprendizaje Automático junto a los cursos de Machine Learning de Stanford y el curso de Detección de Objetos de la Universidad Autónoma de Barcelona, se optó por utilizar uno de los sistemas de detección, tanto para el entrenador como para el clasificador, basados en el trabajo de Viola y Jones [34]

Para la decisión del motor de reconocimiento OCR, se buscó la alternativa de código abierto que diese mejores resultados, y que tuviese un *wrapper* válido para Python, por lo que se optó por Tesseract OCR. Se ha utilizado el diccionario que posee mayor tasa de acierto y mayor tiempo de entrenamiento, que es el diccionario de inglés. Al querer sólo el reconocimiento de los dígitos 0123456789, el propio Tesseract permite pasarle como parámetro la lista de caracteres que debe buscar.

Para la persistencia de la información de la aplicación se han utilizado dos motores de base de datos, uno relacional SQLite, para guardar las imágenes, y otra no relacional, como MongoDB, imágenes para tratar las entidades de los usuarios y administradores.

5. Análisis y Diseño

Tras la superación de la fase preliminar donde pudimos analizar los datos y determinar todas las especificaciones que teníamos que utilizar para nuestra aplicación, se pasa a la fase de análisis y diseño. Al decidir que se iba a desarrollar la aplicación en Python con Django, la aplicación se basa en un patrón MVT, que se explica en detalle más adelante. Vamos a explicar un esquema en la de nuestra aplicación, desde donde el usuario accede a la aplicación, la interfaz de usuario, hasta que los datos de la consulta se guardan en base de datos.

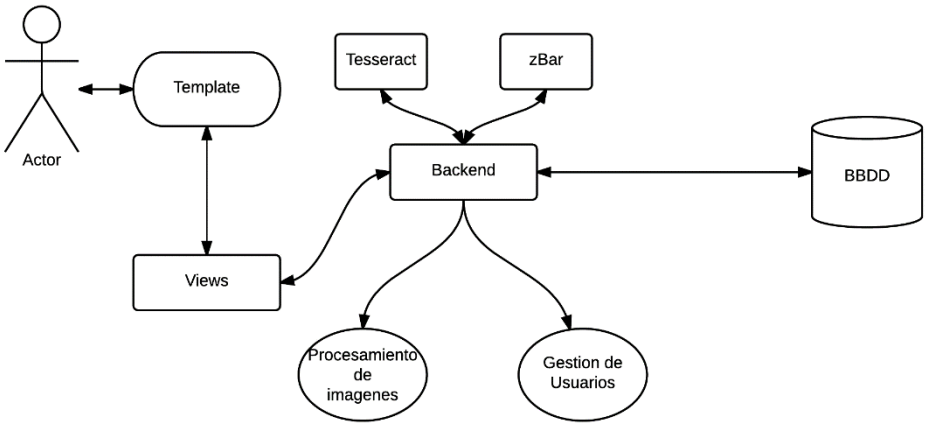


Figura 28: Arquitectura de la aplicación

El usuario lo primero que ve es la interfaz de usuario, en nuestro caso templates, tiene diferentes casos de uso, según las opciones que necesite, aunque algunas de las funcionalidades están restringidas a usuarios registrados en nuestro sistema. La interfaz se relaciona con la capa views, es el enlace entre la lógica de la aplicación o backend y la interfaz. Toda la lógica de la aplicación se encuentra en el backend, donde se produce todo el procesamiento de imágenes y la gestión de usuarios, todos los datos que se necesitan para la aplicación se guardan en la base de datos. El backend es el único punto de la aplicación que se comunica con la base de datos.

5.1. Especificación de requisitos

Para que la aplicación a desarrollar lograse alcanzar los objetivos fijados en un primer momento, se tenían que especificar unos requisitos que consiguieran la máxima funcionalidad para el usuario. El primer requisito importante es la autentificación de usuarios, que incluye el registro y la autenticación. En la zona de usuario se disponen varias funcionalidades, la primera es la carga de imágenes, que incluye de manera transparente al usuario, el tratamiento de imágenes para la detección de la lectura del contador y su número de serie. Después de la lectura se muestran los resultados al usuario para que confirme si los datos obtenidos son correctos y se guarden en base de datos. Por último cada usuario puede ver un histórico de todas sus lecturas anteriores.

A continuación se explican más detalladamente los requisitos planteados. Conviene señalar el hecho de que dentro de estos requisitos, orientados al desarrollo de la aplicación, se contemplan determinadas funcionalidades muy próximas a temas relacionados con la investigación para el reconocimiento de caracteres.

Autentificación de usuarios

Para poder utilizar las partes fundamentales del sistema, los usuarios necesitan registrarse en el sistema, ya que sólo deberían entrar en el sistema usuarios que tengan un contrato con la empresa Madrileña Red de Gas. Una vez verificados los datos, el usuario se puede acreditar (*loguear*) en el sistema para poder guardar sus imágenes y enviar su consumo actual.

Carga y tratamiento de imágenes

Es el requisito fundamental de la aplicación en el que un usuario ya identificado, puede cargar una imagen de su contador. Con la solución propuesta el proceso de tratamiento de imágenes se hace de manera transparente al usuario, es decir, solo tiene que subir la imagen sin necesidad de volver a realizar ninguna otra acción salvo la confirmación de los datos.

Con respecto al tratamiento de imágenes, lo que se realiza es un análisis de la fotografía subida, para detectar las zonas donde se encuentra el código de identificación del contador y el consumo actual. A continuación se procede al reconocimiento de caracteres de ambas zonas, con un diccionario exclusivamente numérico, ya que en ambos casos sólo es necesario reconocer códigos compuestos por dígitos de tal naturaleza.

Confirmar datos de lectura

Una vez que se ha producido la lectura del contador de ambas zonas, los datos del código de identificación y del consumo actual se le muestran al usuario, en vez de guardar los datos directamente en el momento que se obtienen, lo que se hace es que el usuario los pueda confirmar por si se produce algún error de lectura en alguno de los dos casos de identificación.

Histórico

Otro punto importante es que el usuario pueda ver los datos de consumo anteriores para poder comprobar que las lecturas son acordes entre sí y no se produzcan cambios considerables entre ellas. Los datos se clasifican por los meses en el que el usuario ha subido la imagen, lo que permite poder relacionar mejor dichos datos.

5.2. Diseño

Como ya se ha comentado previamente, para el diseño de la aplicación se ha utilizado el framework Django, con el cual es mucho más sencillo aplicar un patrón MVC (Model-View-Controller). En este esquema se separa el Modelo, donde se encuentra la lógica de la aplicación, la Vista, que es la interfaz del usuario con el sistema y el Controlador, que distribuye los casos de uso de la aplicación.

Django es un framework basado en el patrón MVC, que incluye algunas pequeñas modificaciones respecto a aquel, lo que hacen que al final se use el patrón MVT: Modelo, se aplica igual que en MVC; Vista, es el controlador de la aplicación y Plantilla(Template), es la interfaz de comunicación del usuario con el sistema.

El modelo-vista-controlador se divide en las tres capas descritas anteriormente, que se explican a continuación con más en detalle.

- Modelo
    - Contiene el núcleo de las funciones de la aplicación
    - Es independiente del Controlador y la Vista
- Vista
    - Es la interfaz de la aplicación
    - Puede ser notificada cuando hay un cambio en el modelo
- Controlador
    - Reacciona a la petición del cliente, realizando la acción adecuada y creando el modelo

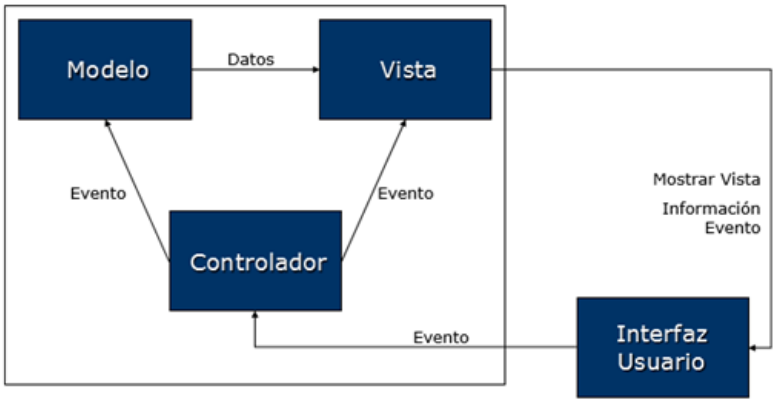


Figura 29: Esquema MVC

Para entender cómo funciona el patrón MVC, se debe entender la división a través del conjunto de los tres elementos de que consta y cómo estos componentes se comunican entre sí, así como con otras vistas y controladores externos al modelo principal. Para ello, es importante saber que el controlador interpreta las entradas del usuario (tanto desde el teclado como del ratón), enviado el mensaje de acción al modelo y a la vista para que se proceda con los cambios que se consideren adecuados.

El modelo, la vista y el controlador deben comunicarse entre sí de una manera estable, de forma que sea coherente con las iteraciones que el usuario necesita realizar durante el uso de la aplicación. Como es lógico, la comunicación entre la vista y el controlador es bastante básica pues están diseñados para operar juntos, si bien los modelos se comunican de una manera diferente, un poco más sutil.

Se requiere algo que comunique el controlador con a la vista, por lo que sí que es necesario el modelo, ya que sólo éste puede llevar a cabo los cambios necesarios al estado actual en el que se encuentran.

Al contrario que el modelo, que puede asociarse a múltiples vistas y controladores, cada vista sólo puede ser asociada a un único controlador, por lo que han de tener una variable de tipo controler que notificará a la vista cuál es su controlador o modelo asignado. De igual manera, el controlador tiene una variable llamada View que apunta a la vista. De esta manera, pueden enviarse mensajes directos el uno al otro y al mismo tiempo, a su modelo.

Al final, la vista es la responsable de establecer la comunicación entre los elementos de patrón MVC. Cuando la vista recibe un mensaje que concierne al modelo o al controlador, lo deja registrado como el modelo con el cual se comunicará y apunta al controlador asignado, para que establezca la vista y así puedan operar conjuntamente. El responsable de deshacer estas conexiones, seguirá siendo la vista, quitándose a sí misma como dependiente del modelo y liberando al controlador.

El patrón MVT es una variante del modelo MVC anteriormente descrito, la variante fundamental es la manera de denominar a la vista y al controlador.

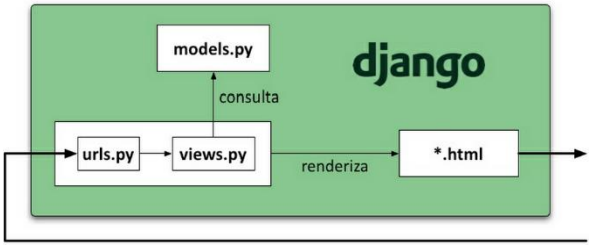


Figura 30: Esquema MVT

- Modelo
    - La capa de acceso a la base de datos
- Vista
    - La capa de la lógica de negocios
    - Puente entre el modelo y la plantilla
- Plantilla (Template)
    - La capa de presentación

La gran diferencia entre un modelo y otro es la manera de interpretar el controlador. En el MVT la vista se interpreta como el controlador de la aplicación, ya que le llegan todas las peticiones de la capa de presentación, siendo el puente entre las plantillas y el modelo. La plantilla es la capa de presentación de este patrón, actúa como si fuese la vista del MVC, ya que se comunica con el controlador que en este caso se llama vista.

6. Implementación

6.1. Tecnologías utilizadas

Para poder alcanzar los objetivos del proyecto, se han utilizado diversas tecnologías que ya estaban implementadas y disponibles con libre acceso. Según lo descrito anteriormente, estas tecnologías se han elegido por su mejor comportamiento en la identificación de los caracteres de acuerdo a las especificaciones del proyecto, a la vez que se han descartado aquellas otras cuyo rendimiento era inferior.

La aplicación está desarrollada en el lenguaje de programación Python [24], ya que entre otras razones se adaptaba perfectamente a la librería fundamental de la aplicación, OpenCV [22], otro argumento que avala su elección que avala su elección es que dicho lenguaje posee mucha documentación asociada, permitiendo tratar problemas más complejos que con otros lenguajes.

Tras la elección del lenguaje de programación lo fundamental para poder desarrollar una aplicación es ver qué tipo se quiere crear, esto es de escritorio o web. Para que fuera mucho más transportable a otros sistemas y se pudiera utilizar en cualquier lugar, decidimos que fuera una aplicación web. Debido a esta elección era necesario elegir un *framework* web para crear toda la aplicación. Aunque existen más opciones, el hecho de haber utilizado Django [10] en otras ocasiones suponía que la curva de aprendizaje iba a ser mucho menor, a la vez que se acoplaba completamente a lo que buscábamos, ya que está basado en el MVC de uso habitual en desarrollo de aplicaciones informáticas. Aparte de esto admite diferentes configuraciones para trabajar en equipo cada uno de los miembros, lo que facilitaba considerablemente la evolución en paralelo de los desarrollos.

Para la capa de presentación, por el hecho de tratarse de una aplicación web fue muy sencillo determinar las tecnologías a utilizar. En efecto, para la inclusión de contenido, tales como imágenes, texto o formularios utilizamos HTML5. Por otra parte, para poder proporcionar una vista mucho más amigable para el usuario y que el aspecto fuese más vistoso se ha utilizado CSS3 y JS (JavaScript). También hemos usado bibliotecas externas tales como Jquery o Bootstrap, que utilizan CSS y JS, con el fin de hacer que la web fuera ajustable a distintos dispositivos a la vez que *responsive.*

Un diseño adaptable o *responsive*, es aquel en el que la visualización del contenido web se adapta de manera automática a distintos dispositivos, resultando de gran utilidad por el hecho de que actualmente existen multitud de tabletas, *smartphones*, ordenadores y cada uno de ellos con un tamaño de pantalla diferente.

Desde el punto de vista del tratamiento de imágenes, la librería más potente de cara a la consecución de los objetivos era OpenCV, ya que posee todas las técnicas anteriormente mencionadas, existiendo sobre ella una documentación que nos permitía ver casos similares para poder captar ideas aplicables a nuestro problema. Imutils [13] es una librería de funciones

que simplifica el proceso de operaciones geométricas básicas en las imágenes, tales como el redimensionado, la rotación o translación, entre otras.

Para poder desarrollar el proyecto de una manera más cómoda utilizamos Bitbucket [4], es un sistema de control de versiones que está basado en Git. Un sistema de control de versiones, sirve para el almacenamiento de la aplicación, de forma que sea posible realizar cambios sobre los elementos almacenados y lo que incluso resulta aún más importante, poder recuperar versiones anteriores, por si se han producido errores, ya que tiene un registro histórico con los cambios introducidos en el proyecto.

Otro punto importante abordado, era cómo transcribir una imagen a texto después de aplicar técnicas de tratamiento de imágenes. Para esto existen los OCR (Optical character recognition) descritos previamente en este proyecto se utilizó Tesseract [7], en nuestro caso adaptado al lenguaje Python utilizado [8]. Nos decidimos por esta opción ya que actualmente está considerado de los mejores OCR libres del mercado [12].

El tratamiento de imágenes, necesita internamente el uso de otras librerías o módulos para su funcionamiento. Para el uso y manipulación de imágenes se ha usado PIL (Python Imaging Library), para ampliar el soporte sobre matrices y vectores, conviene recordar que las imágenes en Python se guardan en matrices, se ha utilizado Numpy. En las primeras versiones de la presente aplicación, para mostrar las imágenes obtenidas y poder ver los resultados de las pruebas, se utilizó Matplotlib.

Para la persistencia de datos se han utilizado dos sistemas, SQLite, que es una BBDD relacional para la gestión de modelos internos de la aplicación, tales como las propias imágenes. Finalmente para la gestión de usuarios y con el fin de guardar los datos de los contadores, se ha usado MongoDB, que es un tipo de BBDD no relacional.

Uno de los objetivos de la aplicación desarrollada es que sea fácilmente trasladable entre sistemas operativos, con lo que se decidió usar Virtualenv [35], la cual es una herramienta que genera entornos virtuales para poder encapsular la aplicación con todos los módulos que necesita. Otra herramienta importante es PIP, sirve para la instalación de los módulos dentro de nuestro sistema.

La detección del código de barras, se podía realizar de dos maneras distintas, la primera y más rudimentaria, era leer los dígitos que aparecen en el contador mediante un OCR pasando los dígitos a texto. La otra manera es su detección mediante la aplicación de los principios matemáticos del código de barras cuya información aparece en las líneas del propio código de barras, para eso se ha utilizado zBar [40] con tal finalidad.

6.2. Proceso de reconocimiento de lecturas

A continuación se describe brevemente el tratamiento de imágenes, ya que al ser un proceso transparente al usuario, a la hora de la carga de la imagen no se pueden visualizar todos los pasos internos que se desarrollan para obtener los resultados finales. En este caso se detallan los tres casos disponibles relativos a los procesos internos: el reconocimiento y lectura del código de barras, el reconocimiento de la zona del contador y su lectura, aplicando en uno de los casos aprendizaje automático y en el siguiente sólo utilizando operaciones de procesamiento de imágenes.

Reconocimiento del código de barras

En la Figura 31, se muestra el resultado del reconocimiento del número de serie a través del código de barras. Se pueden comprobar distintas características, la primera es que el contador no está centrado sobre la imagen, con lo cual el usuario puede hacer la fotografía sin que el único elemento que aparezca en la imagen sea el contador. La segunda es la detención de la zona del código de barras, mediante técnicas de tratamiento de imágenes, la cual se identifica correctamente, a pesar de aparecer otros códigos de barras. Se aprecia en la figura donde la zona del código de barras está identificada con un recuadro de color verde. Tras detectar la región se llama a la función zBar a través de un *wrapper*, devolviendo un número de serie que se ha superpuesto a los resultados en la imagen original.

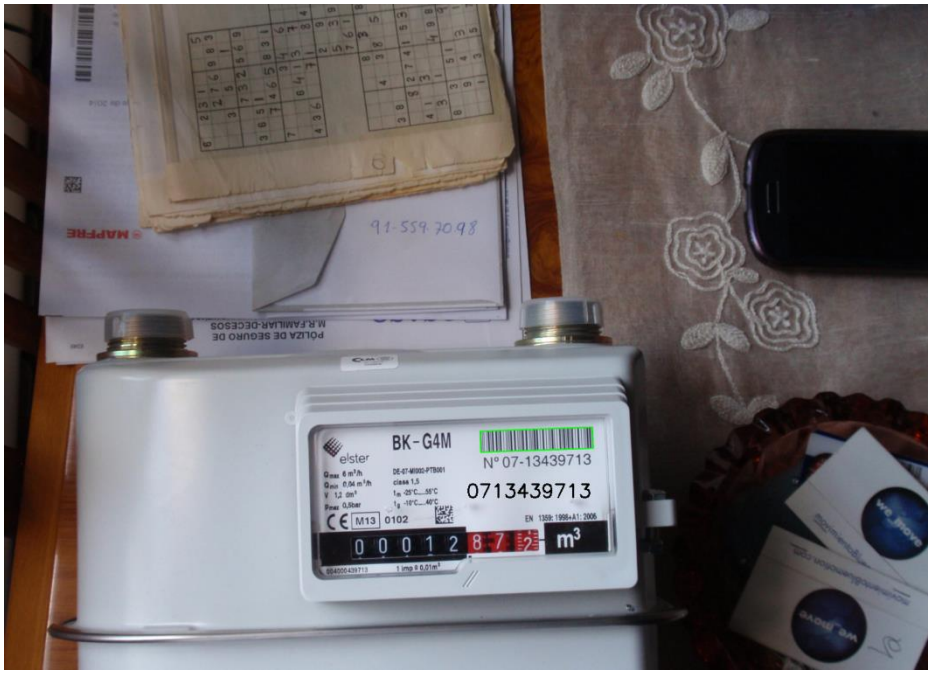


Figura 31: Reconocimiento de código de barras

El proceso completo del reconocimiento del código de barras se puede ver en el Apéndice D.

Reconocimiento de lectura de contador (con aprendizaje automático)

Una de las primeras circunstancias que se pueden apreciar en la Figura 32 es que la zona de la lectura sigue sin estar encuadrada en la fotografía, es decir, la captura de la imagen puede ser tomada sin mucha precisión en cuanto a la zona donde se ubica la región de interés. El punto más importante es que la zona donde se sitúan los dígitos de la lectura se encuadra perfectamente, aunque se encuentren alejados con respecto a la cámara que captura la imagen, ya que debido a las técnicas de aprendizaje automático utilizadas, lo que buscan son regiones de interés similares a las de los ejemplos positivos. Como se puede comprobar los resultados de la lectura de la zona de color negro son correctos. La zona del consumo se encuentra bordeada por un rectángulo verde, hemos superpuesto en la imagen los resultados de la lectura. La empresa nos indicó que la zona de color rojo, en la que se encuentran los decimales no es necesaria para guardarla en los datos del consumo.

Figura 32: Reconocimiento de lectura con aprendizaje automático

Reconocimiento de lectura de contador (sin aprendizaje automático)

La primera diferencia con la imagen anterior es que la Figura 33 está cuadrada con respecto al contador, es decir, no se pueden utilizar imágenes más amplias conteniendo otros elementos o donde el contador no ocupe la parte principal de la captura. Lo que conlleva que el usuario tenga que poner especial cuidado a la hora de realizar la fotografía para que la imagen del contador quede encuadrada. Realmente la empresa planteó como primera aproximación que las imágenes fueran obtenidas de esta manera, ya que aunque la captura requiera mayor atención que en los casos anteriores, los resultados son mejores, ya que la lectura se realiza correctamente en un alto porcentaje de casos.

Figura 33: Reconocimiento de lectura sin aprendizaje automático

7. Conclusiones y trabajo futuro

El objetivo fundamental del proyecto era poder resolver el problema planteado por la empresa Madrileña Red de Gas, que era la lectura del consumo actual del contador, así como su número de serie. La tasa de acierto pretendida era sobre imágenes encuadradas conteniendo el contador, en entornos reales. Esto supone un paso muy importante para el usuario final, ya que éste no tiene que estar pendiente de encuadrar sobre la imagen la parte relativa a la parte frontal del contador.

Una vez conseguido el objetivo principal, se han ido incluyendo mejoras, la más importante es la comentada anteriormente de que se detecta el contador en entornos reales, esto se consiguió mediante técnicas de aprendizaje automático, cuya base fundamental implica un proceso de entrenamiento mediante una serie de imágenes, convenientemente procesadas, de la zona del contador sobre la que se quieren detectar los dígitos tanto del consumo como de la identificación de usuario.

Otro aspecto importante considerado en el proyecto, es que disponga de un patrón MVC. Con dicho patrón hemos conseguido desarrollar una aplicación fácilmente portable a otros sistemas, como por ejemplo disponer de una aplicación nativa para dispositivos móviles, que podría facilitar más la posibilidad de su utilización por parte de un usuario final.

Dos importantes mejoras incluidas hacen referencia por un lado al hecho de que el usuario pueda ver un histórico de sus datos anteriores, para poder seguir sus consumos y si se ha producido algún fallo en relación a las cantidades de meses anteriores; y por otro lado a que la aplicación tenga un diseño adaptable (*responsive*) a cualquier dispositivo móvil, con lo cual antes de desarrollar una aplicación nativa para dispositivos móviles, se puede utilizar desde cualquier dispositivo la propia aplicación web.

Al lograr los resultados esperados inicialmente, hemos alcanzado un punto muy importante para la empresa como es la reducción de los costes, ya que gracias a la aplicación desarrollada, los técnicos no tienen la necesidad de desplazarse a la vivienda de los clientes. Esto conlleva una importante ventaja tanto para la propia empresa, por lo comentado previamente y por el consiguiente ahorro de costes, como para el cliente ya que no tiene por qué estar presente para la lectura de los contadores cuando el técnico se desplace a su domicilio. En los siguientes puntos se concretan los objetivos más importantes conseguidos en el marco de desarrollo del proyecto:

- Resultados acordes a lo esperado por la empresa

- Mejoras con respecto a las ideas iniciales, más concretamente en lo que respecta al uso de técnicas de aprendizaje automático, para una mejor detección de la zona de lectura, imágenes no necesariamente cuadradas, aplicación portable fácilmente o histórico de datos.

- Uso de aplicación web de diseño adaptable con lo que el usuario puede subir una imagen directamente desde un dispositivo móvil.

- Diferencias de resultados sobre imágenes centradas en el contador o imágenes en entornos más amplios.

- Debido a que se han conseguido los resultados esperados inicialmente, hemos alcanzado el objetivo fundamental planteado por la empresa que planteó la necesidad, que es rebajar los costes y facilitar la vida a los usuarios y a la propia empresa a la hora de realizar la lectura, ya que la empresa no tiene que desplazar a un técnico y el usuario no tiene por qué estar presente en la lectura.

Respecto a los trabajos futuros conviene reseñar, que aunque hemos alcanzado los propósitos iniciales y se dispone de una aplicación completamente funcional, que cumple con los casos planteados, siempre existen puntos o aspectos a considerar en el futuro y que se pueden plantear para mejorar la aplicación.

Las primeras mejoras básicas consistirían en incluir una lógica relativa a la lectura del contador que se fundamente en dos cuestiones. La primera seria comprobar que el número de contador y el cliente están vinculados, si no es así la lectura no se debería guardar. Hay dos posibles disyuntivas relativas a este punto o bien que se haya producido un error de lectura en el código de barras o que el cliente haya capturado una imagen de un contador que no es suyo. Una segunda mejora es con respecto a la lógica de la aplicación, consiste en añadir un contexto con respecto a las lecturas, es decir, las lecturas del mes sucesivo siempre tiene que ser mayor o a lo sumo igual que en el mes anterior. Si se logra integrar en la aplicación se podría reducir el número de fallos en el reconocimiento de la lectura [6].

Otro punto importante de cara al futuro de la aplicación es poder portar el proyecto a una aplicación nativa de dispositivos móviles, sobre todo a los sistemas operativos móviles más utilizados concretamente iOS (Apple) y Android (Google). Aunque hemos hecho una pequeña prueba de concepto, que funciona perfectamente en un caso base.

En los últimos días de realización del proyecto, salió al mercado de manera libre también la siguiente versión de OpenCV, en este caso la 3.0, al ser la librería fundamental de la aplicación que hemos utilizado en el tratamiento de imágenes, sería bastante interesante incluirla, entre otros motivos debido a que incorpora nuevas funcionalidades que pueden resultar de mucha ayuda para mejorar la detección de contornos.

Aunque hemos utilizado técnicas de aprendizaje automático, siempre es posible disponer más casos de entrenamiento, para que la tasa de acierto sea mayor, o precisar más los parámetros de entradas para ajustar mejor los resultados. Otro método para mejorar las tasas de acierto es utilizar otras técnicas de aprendizaje automático como pueden ser TBL o HOG. En resumen, las previsiones de futuro se concretan en los siguientes puntos:

- Comprobar que si el cliente y el código de contador no están vinculados, la lectura no debe ser guardada

- Añadir el contexto de la lectura anterior para eliminar fallos en el reconocimiento [6]

- Utilizar OpenCV3.0, lanzada en la fase final de desarrollo del proyecto.

- Utilizar otros métodos de aprendizaje como TBL o HOG para mejorar la tasa de acierto.

Conclusions and future work

The fundamental objective of the project was to be able to solve the problem raised by the Madrileña Gas Network, more specifically the reading of the current consumption of the counter, as well as its identification number or UCPS. The success rate was intended to work on framed images containing the gas meter, in real-world environments. This is a very important step for the end user, since they do not have to be aware of frame on the front of the gas meter.

Once the main goal has been achieved, some improvements have been included, the most important is that the gas meters are detected in real environments, this is achieved through machine learning techniques, whose fundamental basis involves a process of training, using different series of images, conveniently processed, in the area of the gas meter on which you want to detect the consumption digits and the user identification.

Another important aspect considered in the project, is the use of a pattern based on MVC. With this pattern we have succeeded in developing an application easily portable to other systems, such as a native application for mobile devices, which could facilitate the possibility of its use by an end user.

Two major enhancements are that the user is able to see a history of their previous data, in order to be able to continue their consumption and if there has been some fault related to the readings of the past few months; and on the other side the application has an responsive design to any mobile device.

A very important challenge for the company is the cost reduction, achieved thanks to the developed application; the staff does not have to go to the customer's home. This implies a significant advantage for both the company itself, by the previously mentioned resulting cost savings, and for customers that do require their presence during the reading of their gas meters.

In the following points are fleshing out the most important objectives achieved:

- Results that are in line with the expectations of the company

- Improvements with respect to the initial ideas, more specifically regarding the use of machine learning techniques, for a better detection of the reading area, images do not necessarily to be framed over the gas meter, easily portable application and historical data.

- Design and use of a web application customizable, so the user can upload a picture directly from a mobile device.

- Differences in results on images focusing on the gas meter or images in larger environments.

- Results have been achieved as expected. This means that the main objective raised by the company has been achieved. In this regard, costs savings is an important goal..

Regarding the future work it should be noted, that although we have reached the original purposes and available in a fully functional application, that complies with the proposed cases, there are always points or aspects to consider in the future and that can be raised to improve the application.

The first basic improvements would be to include logic relative to the gas meter reading, based on two issues. The first would be to check that the counter number and the customer are linked, if not the reading should not be saved. There are two possible choices in this regard, or that an error has occurred reading the bar code or that the customer has captured an image of a counter that is not owned. A second improvement is based on the logic of the application, by adding a context with respect to the past readings, that is to say, the readings of the succeeding month must always be greater than or at most equal to the previous month. Its integration in the application might reduce the number of faults during the reading recognition [6].

Another important point for the future of the application is to be able to carry out the project to a native application for mobile devices, especially mobile operating systems which are commonly used in iOS (Apple) and Android (Google).

In the last days of completion of the project, another version of OpenCV has been launched, in this case the 3.0 with new interesting functionalities based on image contour detection, which could be of interest taking into account that it was the fundamental library in the application for image processing particularly the image contour detection.

Although we have used automatic learning techniques, it is always possible to increase the number of training patterns, so that could increases the success rate, or further refine the input parameters to adjust better the results. Another method to improve the hit rates is to use other techniques of machine learning as may be TBL or HOG.

In summary, future projections are to be realized in the following points:

- Check that if the customer and the code of the gas meter are not linked, the reading should not be saved

- Add the context of the previous reading to eliminate faults in the recognition [6].

- Use OpenCV3.0, launched in the final phase of development of the project.

- Use other methods of learning as TBL or HOG to improve the success rate.

8. Apéndices

Apéndice A: Instalación de los requisitos

Windows:

En el DVD que acompaña esta memoria se encuentran los instaladores de los requisitos para la puesta en marcha del software necesario para la aplicación. Para ello se ha de descomprimir el archivo RequsitosWindows.zip, abrir una consola en el directorio de ejecutables y seguir los siguientes pasos:

- Instalar python-2.7.9.msi, añadir la opción de "Add Path" en el instalador
- Ejecutar en el cmd "python ez_setup.py" (En la ruta donde tengas los archivos)
- Ejectuar en el cmd "python get-pip.py" (En la ruta donde tengas los archivos)
- Instalar VCForPython27.msi
- Ejecutar en el cmd "pip install numpy"
- Ejectuar en el cmd "pip install matplotlib"
- Instalar opencv-2.4.10.exe en C:/opencv-2
- Copiar C:\opencv\build\python\2.7\x86\cv2.pyd en C:\Python27\Lib\site-packages
- Comprobar que OpenCV se ha instalado correctamente haciendo un import cv2
- Instalar tesseract-ocr-setup-3.02.02.exe en la ruta predeterminada
- Instalar python-tesseract-0.9-0.4.win32-py2.7.exe
- Comprobar que se ha instalado Tesseract haciendo un import tesseract
- Instalar zbar-0.10-setup.exe en la ruta predeterminada
- Instalar zbar-0.10.win32-py2.7_2.msi
- Comprobar que se ha instalado zbar haciendo un import zbar
- Ejectuar en el cmd "pip install image" (Descarga pillow, django e image)
- Ejectuar en el cmd "pip install pymongo"
- Ejectuar en el cmd "pip install django-mathfilters"
- Ejectuar en el cmd "pip install djangorestframework”
- Instalar mongodb-win32-x86_64-2008plus-ssl-3.0.3-signed.msi

Linux:

Copiamos la carpeta RequisitosLinux.zip a nuestro sistema, abrimos una terminal y nos desplazamos hacia la ruta copiada.

- Ejecutamos los siguientes comandos:
- sudo su
- chmod +x dependencies.sh
- ./dependencies.sh
- chmod +x opencv.sh
- ./opencv.sh
- cd leptonica-1.69
- ./configure
- make
- sudo make install
- sudo ln /dev/null /dev/raw1394
- sudo apt-get install automake
- sudo apt-get install build-essential libtool
- sudo apt-get install libzbar-dev
- sudo pip install zbar
- sudo apt-get install tesseract-ocr
- tar xzf tesseract-ocr-3.02.eng.tar.gz
- $ sudo cp tesseract-ocr/tessdata/* /usr/local/share/tessdata
- sudo dpkg -i python-tesseract*.deb
- sudo apt-get -f install
- sudo apt-get install python-pip
- sudo pip install numpy
- sudo pip install nmatplotlib
- sudo pip install image
- sudo pip install pymongo
- sudo pip install djangorestframework
- sudo apt-get install mongodb
- sudo pip install django-mathfilters

Apéndice B: Despliegue de la aplicación

- Primero se ha de ejecutar el software de MongoDB, para ello, la primera vez que se ejecute debemos crear la carpeta donde va a residir nuestra base de datos. Una vez tengamos la carpeta creada, invocamos el software a través de este comando:

  Windows:
  - `C:\mongodb\bin\mongod.exe --dbpath "Ruta de la carpeta creada"`

  Linux & Mac OSX:
  - `mongod --dbpath "Ruta de la carpeta creada" &`

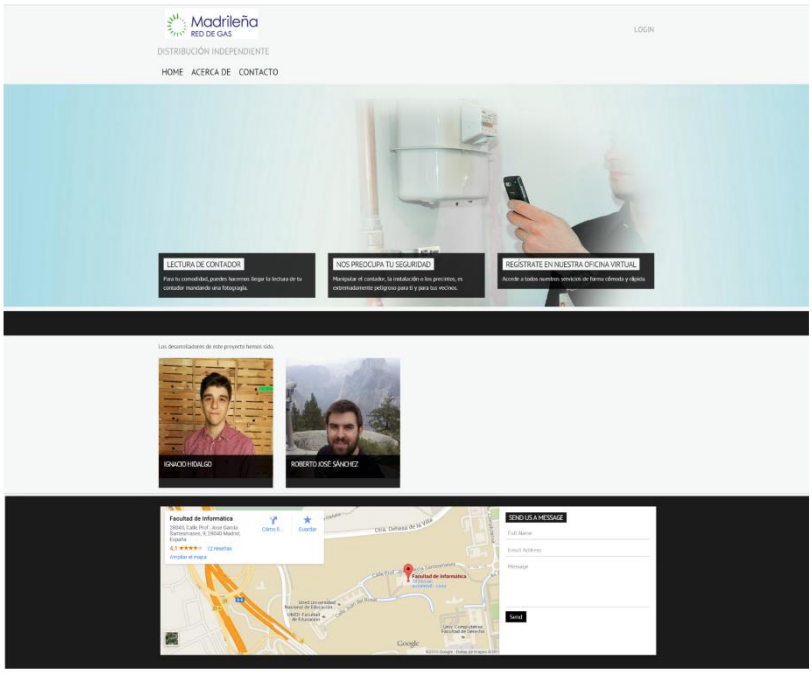
- Tras ejecutar MongoDB, copiamos la carpeta del proyecto que residen en el DVD, llamado "Red de Gas Madrileña –TFG " que acompaña esta memoria, abrimos una consola en ese directorio y ejecutamos:

  - `python manage.py migrate`
  - `python manage.py runserver 8000`

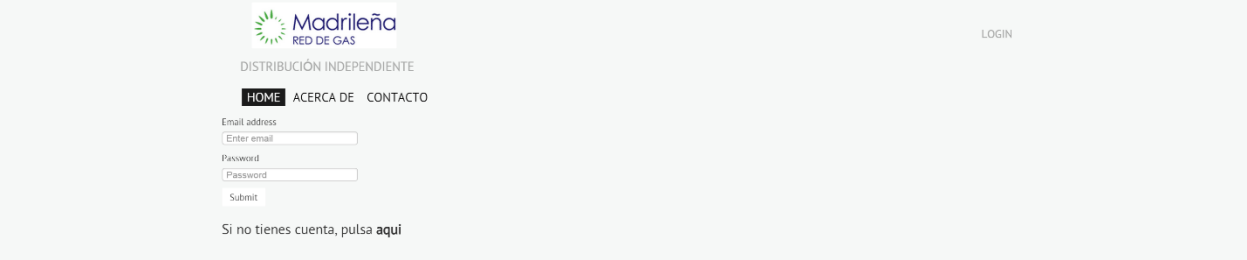
- Ya está corriendo el software, para entrar en la aplicación simplemente, abrir un navegador e introducir en la barra de dirección "localhost:8000" y pulsar Intro.

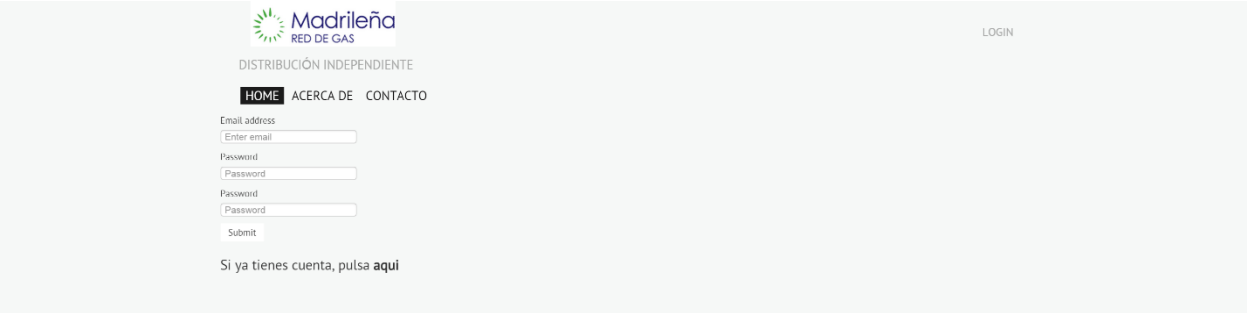
Apéndice C: Uso de la Aplicación

A continuación de detalla un sencillo uso de la aplicación, como es la página inicial, el registro de usuarios, tratar una imagen, guardar los datos o el histórico de imágenes subidas. Con tal propósito se dispone de un servidor donde se puede realizar pruebas y ver el uso de la aplicación, http://52.26.163.81:8000.

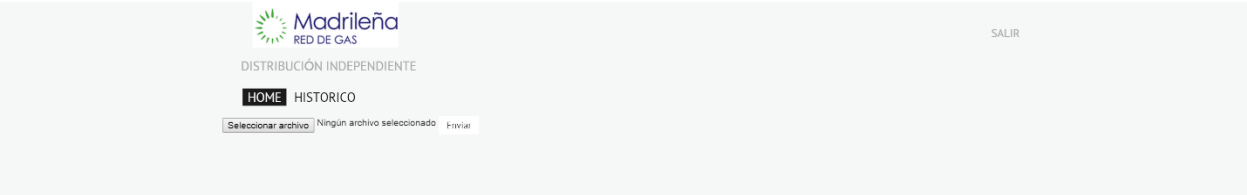


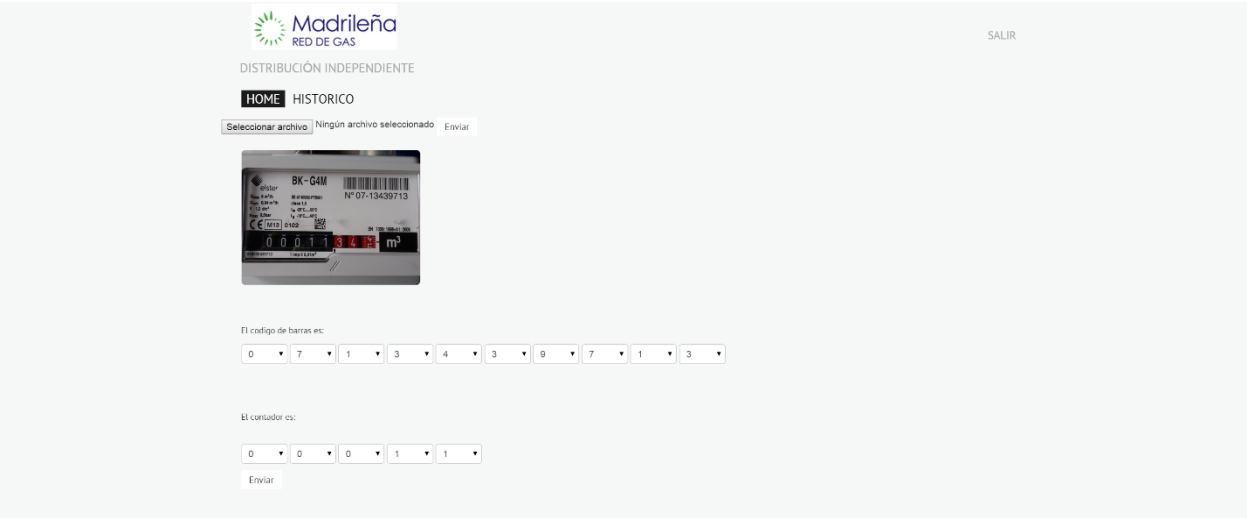
Página inicial, donde se puede ver quien ha realizado la aplicación y datos de la empresa que ha sugerido y propuesto el proyecto.





Páginas de autentificación y registro de usuarios. Para poder loguearse en la aplicación, antes hay que registrarse. Para poder registrarse hay que incluir un correo electrónico y una contraseña.





A continuación se explica el proceso clave para el usuario, sólo tiene que subir una imagen, cuando la aplicación procese la imagen y proporcione los datos procesados, el usuario posee la capacidad de modificarlos por si se ha producido algún error.

Este es el último punto en el que el usuario puede ver tanto las imágenes que ha subido como la fecha de la subida.

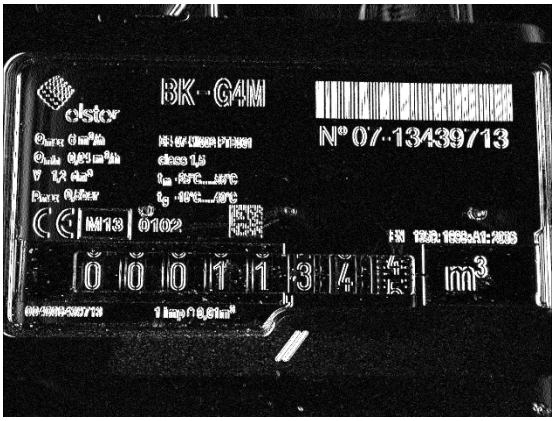
Apéndice D: Detección del número de serie del contador

El proceso relativo al reconocimiento de la región conteniendo el código de barras se muestra en la siguiente secuencia gráfica de ejemplos:

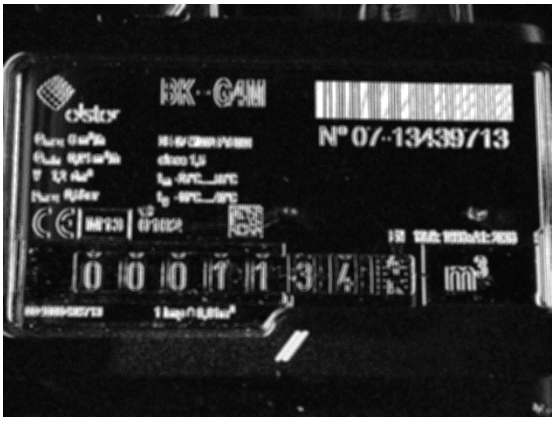
Imagen Original

Escala de grises

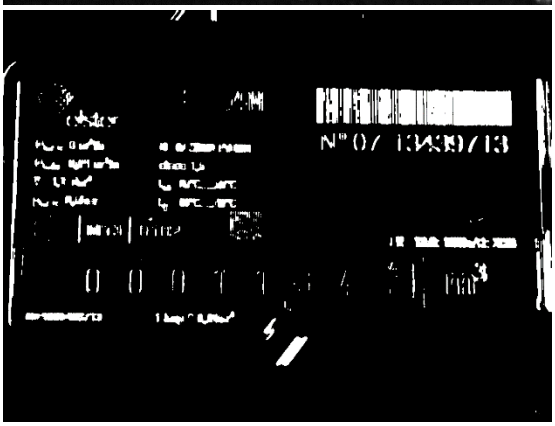
Sobel (Detección de líneas verticales)



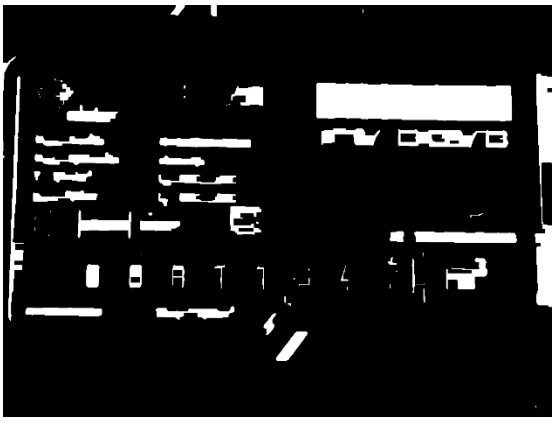
Suavizado



Umbralización

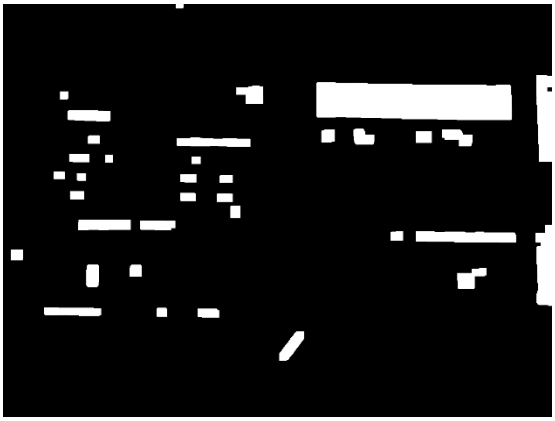


Extracción morfológica (Rectángulos)



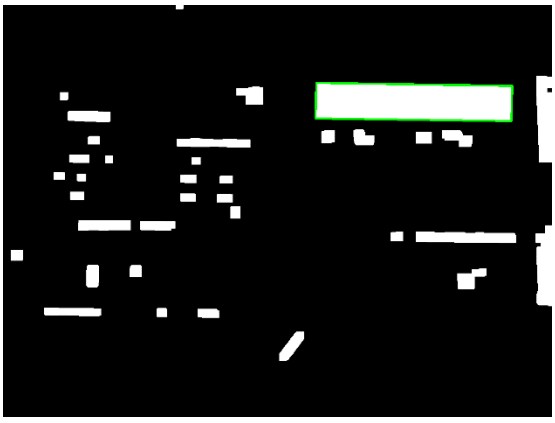
Erosión

Dilatación



Detección de contornos

Obtener rectángulo de mayor área



Resultado

Tras obtener la región, cortamos la imagen quedando:

El código de barras es un EAN18, que se convierte del formato de imagen de OpenCV al formato de imagen de la librería PIL de Python, después a un formato bruto (RAW) y por último se transfiere mediante el *wrapper* a zBar, obteniendo el siguiente resultado:

4150000007134397l3

El código CUPS lo componen sólo los últimos 9 dígitos, por lo cual se almacena sólo esa información.

Apéndice E: Detección de la región de la lectura (sin aprendizaje)

En la siguiente secuencia gráfica se muestra el proceso de detección de la región de la lectura sin aprendizaje.

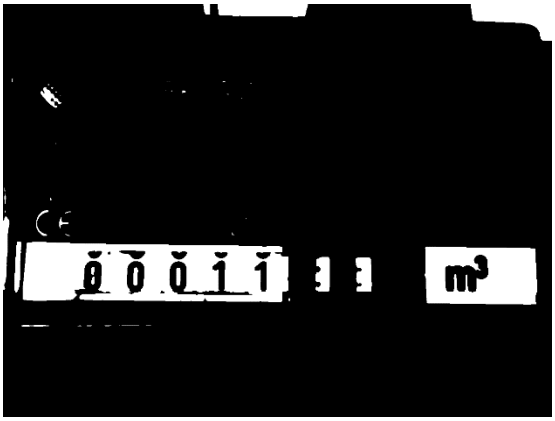
Imagen Original

Escala de grises

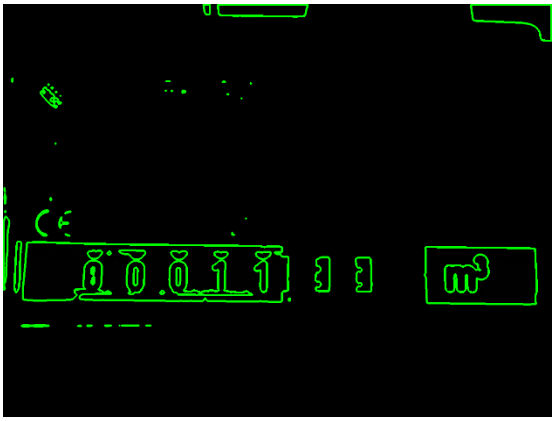
Inversa

Suavizado

Umbralización



Detección de contornos



Contorno de mayor area

Región de interes

Inversa

Recorte de bordes

Thresholding – Imagen Final

La imagen final se envía a través del *wrapper* para su análisis mediante Tesseract-OCR. Los parámetros se corresponden con el diccionario de inglés por defecto, el modo de página de bloque simple y la lista de caracteres posibles “0123456789”. El resultado es el siguiente:

```
C:\POC>python POCs.py ./images/prueba1.png
El codigo del contador es 0713439713
La lectura del contador es: 00011
```

Apéndice F: Detección de la región de la lectura (con aprendizaje)

En la siguiente secuencia gráfica se muestra el proceso de detección de la región de la lectura con aprendizaje.

Imagen Original

Imagen extraída del detector

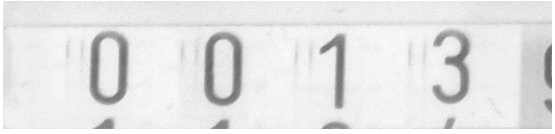
Escala de gises

Invertida

Suavizado

Umbralización

Región de interés



Eliminamos bordes

Thresholding / Imagen Final

La imagen final se envía mediante el *wrapper* para su análisis con Tesseract-OCR. Los parámetros se corresponden con el diccionario de inglés por defecto, el modo de página de bloque simple y la lista de caracteres posibles “0123456789”. El resultado es el siguiente:

```
C:\POC>python POC.py ./images/prueba4.png
El codigo de contador es: 0713439713
La lectura del contador es: 0013
```

Apendice G: Entrenamiento basado en Cascada Haar

Este entrenamiento está basado en la técnica propuesta en [27].

- Copiamos la carpeta de Entrenador, y nos desplazamos a ella mediante un terminal
- Colocamos los casos positivos en la carpeta ./positive_images y creamos una lista con:
  - find ./positive_images -iname "*.jpg" > positives.txt
- Colocamos los casos negativos en la carpeta ./negative_images y creamos una lista con:
  - find ./negative_images -iname "*.jpg" > negatives.txt
- Creamos los ejemplos con bin/createsamples.pl y los guarda en la carpeta. /samples:
  - perl bin/createsamples.pl positives.txt negatives.txt samples 1500\
    "opencv_createsamples -bgcolor 0 -bgthresh 0 -maxxangle 1.1\
    -maxyangle 1.1 maxzangle 0.5 -maxidev 40 -w 80 -h 40"
- Compilamos el archivo mergevec.cpp que sen encuentra en la carpeta ./src :
  - cp src/mergevec.cpp ~/opencv-2.4.9/apps/haartraining
  - cd ~/opencv-2.4.9/apps/haartraining
  - g++ `pkg-config --libs --cflags opencv | sed 's/libtbb\.dylib/tbb/'`\
    -I. -o mergevec mergevec.cpp\
    cvboost.cpp cvcommon.cpp cvsamples.cpp cvhaarclassifier.cpp\
    cvhaartraining.cpp\
    -lopencv_core -lopencv_calib3d -lopencv_imgproc -lopencv_highgui -lopencv_objdetect
- Usamos el ejecutable compilado mergevec para juntar los ejemplos en un solo archivo colocado en ./samples :
  - find ./samples -name '*.vec' > samples.txt
  - ./mergevec samples.txt samples.vec

- Comenzamos el entrenamiento utilizando la función del OpenCV opencv_traincascade, y guardamos los resultados en./classifier:
    - opencv_traincascade -data classifier -vec samples.vec -bg negatives.txt\
      -numStages 20 -minHitRate 0.999 -maxFalseAlarmRate 0.5 -numPos 1000\
      -numNeg 600 -w 80 -h 40 -mode ALL -precalcValBufSize 1024\
      -precalcIdxBufSize 1024

9. Bibliografía

[1] a9t9, The Best Online OCR Software for Converting Images to Text, (2015).

[2] Abby.co.il, History of OCR, (n.d.).

[3] M.V. Antonio López Peña, Ernest Valveny, Maria Vanrell, Detección de Objetos, Universitat Atónoma de Barcelona, Barcelona, 2015.

[4] Bitbucket.org, Git and Mercurial code management for teams, (2015).

[5] S.-L. Chang, L.-S. Chen, Y.-C. Chung, S.-W. Chen, Automatic License Plate Recognition, IEEE Trans. Intell. Transp. Syst. 5 (2004) 42–53.

[6] M. Chouiten, P. Schaeffer, Vision based mobile Gas--Meter Reading, (n.d.).

[7] Code.google.com, tesseract-ocr - An OCR Engine that was developed at HP Labs between 1985 and 1995... and now at Google. - Google Project Hosting, (2015).

[8] Code.google.com, python-tesseract - python wrapper class for tesseract OCR (Linux & Mac & Windows) - Google Project Hosting, (2015).

[9] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: 2005: pp. 886–893.

[10] Djangoproject.com, The Web framework for perfectionists with deadlines | Django, (2015).

[11] Es.wikipedia.org, Reconocimiento automático de matrículas, (2015).

[12] Es.wikipedia.org, Tesseract OCR, (2015).

[13] GitHub, jrosebr1/imutils, (2015).

[14] J.R.H. González, J.P.R. Villaverde, J.A.G. de Mesa, Sistema de reconocimiento óptico de caracteres (OCR) con redes neuronales, Dep. Ciencias La Comput. Univ. Alcalá Henares. (1996).

[15] R.C. Gonzalez, P.A. Wintz, Digital image processing, Addison-Wesley, 1987.

[16] S.D. H, Apparatus for reading, (1953).

[17] D.-C. He, L. Wang, Texture Unit, Texture Spectrum, And Texture Analysis, IEEE Trans. Geosci. Remote Sens. 28 (1990) 509–512.

[18] M. Katona, L.G. Nyúl, A novel method for accurate and efficient barcode detection with morphological operations, in: 2012: pp. 307–314.

[19] Madrilena.es, Madrileña Red de Gas, (2015).

[20] A. Nodari, I. Gallo, A Multi-Neural Network Approach to Image Detection and Segmentation of Gas Meter Counter., in: 2011: pp. 239–242.

[21] T. Ojala, M. Pietikainen, D. Harwood, Performance evaluation of texture measures with classification based on Kullback discrimination of distributions, in: Proc. 12th Int. Conf. Pattern Recognit., IEEE Comput. Soc. Press, 1994: pp. 582–585.

[22] Opencv.org, OpenCV | OpenCV, (2015).

[23] G. Pajares Martinsanz, J.M. De la Cruz García, Visión por computador, imágenes digitales y aplicaciones, (2001).

[24] Python.org, Welcome to Python.org, (2015).

[25] M.T. Qadri, M. Asif, Automatic Number Plate Recognition System for Vehicle Identification Using Optical Character Recognition, in: 2009 Int. Conf. Educ. Technol. Comput., IEEE, 2009: pp. 335–338.

[26] A. Ripardo de Alexandria, P. Cesar Cortez, J. Hebert da Silva Felix, T. Menezes de Oliveira, A. Maia Girao, J. Batista Bezerra Frota, et al., An OCR System for Numerals Applied to Energy Meters, IEEE Lat. Am. Trans. 12 (2014) 957–964.

[27] N. Seo, Tutorial: OpenCV haartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features), (2014).

[28] R. Smith, An Overview of the Tesseract OCR Engine., in: 2007: pp. 629–633.

[29] R. Smith, Tesseract OCR Engine, in: G. Inc. (Ed.), Google Inc., 2007.

[30] Sourceforge, jTessBoxEditor, Sourceforge. (2015).

[31] G. Tatwchek, Beading machine, (1935).

[32] B. Thorne, R. Grasset, R. Green, Using Python in Computer Vision: Performance and Usability, (n.d.).

[33] M. Vanetti, I. Gallo, A. Nodari, GAS meter reading from real world images using a multi-net system, Pattern Recognit. Lett. 34 (2013) 519–526.

[34] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: 2001: p. -–511.

[35] Virtualenv.pypa.io, Virtualenv, (2015).

[36] H.P. W, Statistical machine, (1933).

[37] Wikipedia, Visión artificial - Wikipedia, la enciclopedia libre, (2015).

[38] N.J. Woodland, S. Bernard, Classifying apparatus and method, Google Patents. (1952).

[39] K. Xie, Automatic Utility Meter Reading, (2010).

[40] Zbar.sourceforge.net, ZBar bar code reader, (2015).